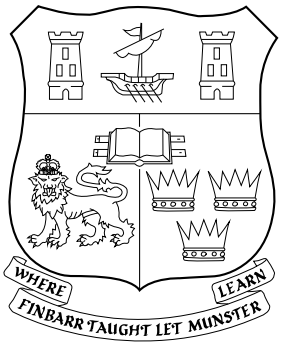
Constraint Programming for Optimization Under Uncertainty in Inventory Control

ROBERTO ROSSI



A Thesis Submitted to the National University of Ireland

in Fulfillment of the Requirements for the Degree of

Doctor of Philosophy in the Faculty of Science.

October, 2008

Research Supervisors: Dr. S. Armagan Tarim,
Dr. Brahim Hnich, and
Dr. Steven D. Prestwich.
Head of Department: Prof. James Bowen

Department of Computer Science,
National University of Ireland, Cork.

Contents

Abstract

Constraint Programming (CP) is a programming paradigm where relations between variables can be stated in the form of constraints. CP features discrete domains and global constraints. Global constraints capture interesting substructures of a problem, encapsulate dedicated inference algorithms based on feasibility and/or optimality reasoning, and provide information to the search process on the most viable course. Stochastic Constraint Programming (SCP) is a novel framework that generalizes CP to stochastic problems, allowing both to model and solve this class of problems by using any available existing CP solver. Although this framework proves to be extremely flexible in terms of modeling power, its current implementation does not scale well.

In order to enhance this framework, in this dissertation we propose a general extension for SCP: global chance-constraints. In contrast to global constraints, which represent relations among a non-fixed number of decision variables, global chance-constraints represent relations among a non-fixed number of decision variables and stochastic variables. Nevertheless, as global constraints do, global chance-constraints encapsulate dedicated inference algorithms based on feasibility and/or optimality reasoning and may provide information to the search process. We call optimization-oriented global chance-constraints those global chance-constraints performing optimality reasoning.

We applied global chance-constraints encapsulating dedicated inference algorithms based on feasibility and/or optimality reasoning to problems in the area of stochastic inventory control. Our computational experience shows that global chance-constraints let us model and solve to optimality problems that could not or could be only approximately solved by other existing approaches. It also shows that filtering based on optimality reasoning is extremely effective for this class of problems.

Roberto Rossi, Cork Constraint Computation Centre, University College Cork, College Road, Cork, Ireland.

Declaration

This dissertation is submitted to University College Cork, in accordance with the requirements for the degree of Doctor of Philosophy in the Faculty of Science.

Parts of this dissertation are the results of collaboration with Dr. Armagan Tarim, Dr. Brahim Hnich and Dr. Steven D. Prestwich. I declare that I have made a substantial contribution to this work and this dissertation is composed by myself. This dissertation has not been submitted to any other university or higher education institution, or for any other academic award in this university. Where use has been made of other people's work, it has been fully acknowledged and referenced.

The papers contained in this dissertation, or upon which this dissertation is based, have either been published in, been accepted for publication at, or been submitted for publication to, as indicated, at reviewed journals, conferences or workshops.

The papers have not been edited except to fix typographical or spelling errors and to update references. They have, however, been reformatted for this dissertation and thus floating objects, such as figures and tables, may have moved about with respect to their surrounding text.

Paper I S. A. Tarim, B. Hnich, R. Rossi, and S. Prestwich, "A Global Chance-Constraint for Stochastic Inventory Systems under Service Level Constraints", Constraints, an International Journal, Vol. 13(4):490-517, 2008

Paper II R. Rossi, S. A. Tarim, B. Hnich and S. Prestwich, "Computing Replenishment Cycle Policy under Non-stationary Stochastic Lead Time",

submitted for possible publication to the International Journal of Production Economics

Paper III R. Rossi, S. A. Tarim, B. Hnich, and S. Prestwich, "Cost-based filtering for stochastic constraint programming", *In proceedings of The 14th International Conference on Principle and Practice of Constraint Programming (CP-2008), Sep. 14-18, 2008 - Sydney, Australia*, Lecture Notes in Computer Science, Springer-Verlag, LNCS 5202, pp.235-250, 2008

Paper IV R. Rossi, S. A. Tarim, B. Hnich, and S. Prestwich, "Cost-based Filtering Techniques for Stochastic Inventory Control under Service Level Constraints", Constraints, an International Journal, forthcoming, 2009. Extended version of: S. A. Tarim, B. Hnich, R. Rossi, and S. Prestwich, "Cost-Based Filtering for Stochastic Inventory Control", *Recent Advances in Constraints: 11th Annual ERCIM International Workshop on Constraint Solving and Constraint Logic Programming, CSCLP 2006 Caparica, Portugal, June 26-28, 2006 Revised Selected and Invited Papers*, Lecture Notes in Computer Science, Springer-Verlag, LNCS 4651, pp.169-183, 2007

Paper V R. Rossi, S. A. Tarim, B. Hnich and S. Prestwich, "Constraint Programming for Stochastic Inventory Systems under Shortage Cost", submitted for possible publication to the European Journal of Operational Research. Extended version of: R. Rossi, S. A. Tarim, B. Hnich, and S. Prestwich, "Replenishment Planning for Stochastic Inventory Systems with Shortage Cost", *In proceedings of The Fourth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR 07), May 23-26, 2007, Brussels, Belgium*, Lecture Notes in Computer Science, Springer-Verlag, LNCS 4510, pp.229-243, 2007

Some of these papers are reprinted with the permission of the publishers.

I will provide more details concerning joint work. In Chapter 2 (Paper I), I proposed the original idea of *global chance-constraint*, I developed the related software and I have written most of the articles. In Chapter 3 (Paper II), the motivation comes from Dr. Tarim, who first suggested to consider a stochastic lead time in our model. We developed together the mathematical model and I implemented the related software on my own. Again I have written most of the articles with the invaluable support of Dr. Hnich. The motivation for Chapter 4 (Paper III) comes from Dr. Tarim and myself. In particular the use of Jensen's inequality was firstly suggested by Dr. Tarim. Nevertheless I proposed the idea of *optimization-oriented global chance-constraint* in which this inequality is used to generate bounds and perform filtering. I also identified the motivation problems and I implemented the software for running the experiments. Chapter 5 (Paper IV) presents ideas proposed by Dr. Tarim, Dr. Hnich and myself. Specifically I proposed the filtering strategy presented in Section 5.3.1. I implemented all the related software and also a graphical interface for visualizing results. I wrote most of the article. Finally, Chapter 6 (Paper V) presents ideas that I originally proposed, implemented, and put together in a conference paper first, and then in a journal article. Also in this case Dr. Tarim, Dr. Hnich and Dr. Prestwich made important comments and had an active role in the writing of the research articles.

Roberto Rossi
July 2008.

Dedication

To my family.

Acknowledgements

Roberto Rossi is supported by Science Foundation Ireland (SFI) under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain Research (CTVR) and Grant No. 05/IN/I886.

After the formal acknowledgements, I would like to thank in this small part of my dissertation all the people that contributed to my personal and professional development during my PhD.

I will list people in chronological order to be fair with everyone. I want to thank Michela Milano, who sent me to Cork three years ago and made me discover this incredible community of researchers. I am very grateful to the people in Cork who hosted me for my first six months and believed in me letting me stay for the following three years: Armagan Tarim and Brahin Hnich. I will never forget our first lunch at Mercury Lounge (now closed down) and Gusto (still doing great business with 4C) and I will never forget our brainstorming sessions at 4C. I want to thank my (official) supervisor Steve Prestwich, who has been extremely supportive during my all PhD and in the last 2 years in particular, when both Armagan and Brahim left for the sunny Turkey. I want also to thank all the people of the Cork Constraint Computation Centre (4C) with who I had fruitful discussions and great laughters during pool tournaments, barbecues and other nice events. 4C has been an incredibly stimulating place for research. A special thank goes to Eleanor, Linda and Caitriona for all the administrative support. I wish to thank the people in Bell Labs Ireland for hosting me several times in Dublin and for the time of work and leisure we had. I am grateful to my family that did everything was possible to make me study and to let me get where I am now. Finally I am grateful to Lauren, for sharing with me in the last two years all the good and bad things of our life.

Chapter 1

Introduction

1.1 Preliminaries

In this section we firstly provide the motivations for the work presented in this dissertation; secondly we briefly state the topic discussed in this dissertation; and finally we discuss the structure of the rest of this chapter.

1.1.1 Motivations

Many computational problems can be described in terms of restrictions imposed on the set of possible solutions, and Constraint Programming is a problem-solving technique that works by incorporating those restrictions in a programming environment. It draws on methods from combinatorial optimization and Artificial Intelligence, and has been successfully applied in a number of fields from scheduling, computational biology, finance, electrical engineering and Operations Research through to numerical analysis.

Constraint Programming has been extremely successful in the field of deterministic production planning and scheduling [47]. The commercial success of off-the-shelf tools such as ILOG Scheduler [49] is remarkable.

Nevertheless, real-life management decisions are usually made in uncertain environments. Random behavior such as the weather, lack of essential exact information such as the future demand, incorrect data due to errors in measurement, and vague or incomplete definitions, exemplifies the theme of uncertainty in such environments.

In this work we aim to investigate the application of Constraint Programming to decision problems under uncertainty and in particular to production/inventory control problems. Having an effective means to handle these problems is a key to profitability for retail business, which is particularly affected by uncertainty. Supply chains are plagued by uncertainty associated with customers' demand, lead-times, suppliers' capacity, and so forth. We now provide some evidence of the impact that uncertainty has on retail and on the importance of having state-of-the-art decision support systems for hedging against it.

Retail replenishment[†] *is a high-value activity. According to the US Commerce*

[†]The process of moving or re-supplying inventory from a reserve storage location to a primary

Department, $1.1 trillion in inventory supports $3.2 trillion in annual US retail sales. This inventory is spread out across the value chain, with $400 billion at retail locations, $290 billion at wholesalers or distributors and $450 billion with manufacturers. This is a colossal amount of capital tied up in inventory [...]. Improving distribution centre efficiency of just a few percentage points through advanced automation and real-time replenishment may deliver significant savings and require less capital to be tied up in inventory.[‡]

Table 1.1 shows inventory as a percentage of total assets for some major industries. It appears that such an amount of inventory should significantly reduce

Industry	**Inventory relative to total assets**
Automotive dealers and service stations (retail)	53.81%
Apparel and accessory stores	41.14%
Building materials, garden supplies and mobile home dealers (retail)	40.09%
Food stores	33.52%
Electrical and electronic equipment	19.57%
Total construction	17.20%

Table 1.1: Inventory as a percentage of total assets for some major industries. Data source: Internal Revenue Service, U.S. Treasury Department, Statistics of Income, 1977; Corporate Income Tax Returns (Washington, DC: Government Printing Office, 1982), pp. 27-34.

the probability of stock-out[§] at retail level. In fact, many surveys reveal that what happens in reality is that a high percentage of shoppers, on average, fail to find products in stock. Stock-out events for many firms represent a significant portion of all retail sales. Even if some of these events are actually recouped via alternative products, still the lost sales faced by these firms remain high. Obviously this is seriously affecting both retail margins and customer satisfaction. Overstocks[¶], on the other hand, can be just as damaging financially to the organization. Nowadays no retailer can afford to tie up capital unnecessarily in inventory, or risk lost sales and dissatisfied customers due to stock-outs. However, current practices put

picking location, or to another mode of storage in which picking is performed.

[‡]"The Future of Retail Replenishment", Manhattan Associates ©, 2006, http://www.manh.com/library/MANH-TechVis_Whitepaper.pdf

[§]When at a given moment in a given inventory there is not the quantity of a part or a product that is demanded. A stock-out occurs in a distribution center when there are orders that can not be filled within their due date.

[¶]To stock more products than strictly necessary or desirable.

in place by firms seem unable to produce a balanced situation where the right good is in the right place at the right time. High stockout levels in retail settings prove to be the norm, rather than the exception. As a study conducted in 1996 by the Andersen Consulting Group — today known as Accenture — revealed, on a typical afternoon in a typical US supermarket, 8.2% of items are out of stock, and this number is nearly doubled for items that are advertised. In 3.4% of stockouts, consumers refuse to buy an alternative and often take their business to the competition. The costs of stockouts in US supermarkets alone are estimated at $7-12 billion of sales. This example illustrates the drastic consequences of stockouts, and underlines the importance of properly managing inventory investments by means of sound modeling techniques and advanced decision support systems.

In the last few decades the Operations Research community developed a large amount of lore for decision making under uncertainty. Stochastic Programming (see Sengupta [78], Vajda [95], Kall and Wallace [54]) has been widely and successfully applied to problems from the retail world.

In contrast to what has happened in the Operations Research community with Stochastic Programming, only recently the Constraint Programming community has started to formalize general approaches that employ Constraint Programming for optimization under uncertainty. The probabilistic CSP framework [29] has been one of the first work in this direction. Relevant works are also Partial CSPs [36] and Soft CSPs [13]. Nevertheless, none of these approaches is as general as the techniques employed in Stochastic Programming. The very first step towards the integration of Constraint Programming and Stochastic Programming was made by Walsh, who introduced Stochastic Constraint Programming [98] a novel framework able to fully represent the stochastic nature of decision problems under uncertainty. Stochastic Constraint Programming is still a young field, and only recently a general purpose modeling and solution framework was proposed for stochastic constraint programs in [91]. There are still several issues open both in terms of expressiveness of the framework and of efficiency of the current solution methods available. Applications to real world problems are also very limited. In this sense Stochastic Constraint Programming is indeed an interesting “green field” for research.

We claim that Stochastic Constraint Programming may bring significant ben-

efits in the field of stochastic inventory/production control. The results we present in this work fully support this thesis. In fact we propose Stochastic Constraint Programming approaches for inventory control that are:

- *more accurate than other existing approaches in the literature*. The quality of the solution found is improved significantly, i.e. costs are reduced, and the expected cost predicted is closer to that realized in practice;
- *more effective in terms of computational performance*. Our Constraint Programming reformulations proved to be orders-of-magnitude more efficient than other approaches in the literature;
- *more effective in terms of expressiveness*. Constraint programming reformulations are particularly compact and, as shown in [92], require fewer constraints and decision variables than other existing approaches in the literature.

As discussed above, large amount of capital are invested in inventories by firms. Having more effective, accurate and efficient approaches to inventory optimization is therefore desirable. The research presented in this dissertation tries to pursue these objectives.

Topic. *In this dissertation we investigate the application of Stochastic Constraint Programming techniques and in particular of global chance-constraints, a novel modeling concept introduced here, in the area of stochastic inventory control. We implemented global chance-constraints encapsulating dedicated inference algorithms based on feasibility and/or optimality reasoning. Our computational experience shows that global chance-constraints let us model and solve to optimality problems that could not or could be only approximately solved by other existing approaches. It also shows that filtering based on optimality reasoning is extremely effective for this class of problems..*

1.1.2 Structure

The rest of this chapter is structured as follows:

- in Section 1.2, we provide the relevant formal background in Dynamic Programming, Constraint Programming, Stochastic Constraint Programming, and inventory control;
- in Section 1.3, we discuss the relevant literature in Stochastic Constraint Programming and stochastic inventory control; then we also discuss existing techniques for integrating Operations Research and Constraint Programming approaches in Combinatorial Optimization;
- in Section 1.4, we summarize the content of this dissertation, we state at a high level our contributions, and finally for each of the following chapters we list the respective contributions in details;
- in Section 1.5, we discuss possible future research directions. Specifically, for each of the following chapters we discuss which questions remain open and which directions may be interesting to follow in the future research;
- in Section 1.6, we draw conclusions.

The general structure of this dissertation will be further discussed in Section 1.4.

1.2 Formal background

In this section we discuss the relevant formal background in the areas of Dynamic Programming (Section 1.2.1), Constraint Programming (Section 1.2.2) and Stochastic Constraint Programming (Section 1.2.3), a framework that employs Constraint Programming for solving decision problems under uncertainty. Finally we discuss relevant topics in stochastic inventory control (Section 1.2.4).

1.2.1 Dynamic Programming

This section is mainly based on [33].

Dynamic Programming (DP) is an optimization procedure that solves optimization problems by decomposing them into a nested family of subproblems. The core of DP is the *principle of optimality* [8, 25].

In DP a problem P is associated with a *state space graph* $SG = (S, T)$ where each element of the vertex set S is a state and each element of the arc set T represents a feasible transition between two states. The original problem is solved by solving a shortest path problem in the state space graph from an initial state to a final state (boundary condition)[‖]. If the original problem is NP-hard, the corresponding state space graph will have an exponential number of nodes.

Consider a discrete system defined on n steps. Each step is characterized by:

- a final state s_k that represents the system at the end of step k. $s_k \in S_k$, where S_k is the set of feasible states at the end of step k.
- a decision variable x_k that represents a decision taken at step k. $x_k \in X_k$, where X_k is the set of feasible decisions that could be taken at step k.
- a cost/profit function $p_k(s_k, x_k)$ representing the cost/profit achievable in step k if s_k is the final state and x_k the decision considered.
- a state transition $t_k(s_{k-1}, x_k)$ that leads the system toward the state $s_k = t_k(s_{k-1}, x_k)$.

[‖] This definition of DP is restrictive, but sufficient for the discussion in this work.

Without loss of generality we will here refer to minimization problems. Optimization problems aim at finding the set of *optimal values* to be assigned to decision variables such that the following objective function is minimized:

$$z = \min \left\{ \sum_{k=1}^{n} p_k(s_k, x_k) \right\}.$$

To determine the value of z, DP solves a set of problems $i = 1, \ldots, n$, each corresponding to a system composed by i steps and characterized by the state s_i at the end of step i. The recursive formulation of the cost function at step i is:

$$f_i(s_i) = \min_{x_i \in X_i} \left\{ \min_{s_i \in S_{i-1}} \{ f_{i-1}(s_{i-1}) + p_i(s_i, x_i) \} \right\}$$

where $s_i = t_i(s_{i-1}, x_i)$. In addition, we have the following boundary condition:

$$f_1(s_1) = \min_{x_1 \in X_1} \{ p_1(s_1, x_1) \}$$

where $s_1 = t_1(s_0, x_1)$.

DP is based on the *principle of optimality* [8] stating that an *optimal policy* is such that given whatever state s_i, and the decision x_i, the decisions $x_1, \ldots, x_{i-1}$ corresponding to the remaining steps constitute an optimal policy w.r.t. the state s_{i-1} resulting from the decision taken at step i.

DP is often applied to problems requiring a sequence of interrelated decisions, and has been applied to solve a wide variety of combinatorial optimization problems, as well as optimal control problems. Recently, effective hybrid optimization techniques involving DP and Constraint Programming have been proposed in [33]. In Chapters 5 and 6, we develop similar hybrid techniques in order to efficiently solve combinatorial optimization problems for inventory control. In the next section we formally introduce Constraint Programming.

1.2.2 Constraint Programming

This section is mainly based on [1].

Let v be a variable. The *domain* of v is a set of values that can be assigned to

v. In what follows we will restrict our attention to *finite* domains. Consider a finite set of variables $\mathcal{V} = \{v_1, v_2, \ldots, v_k\}$, where $k > 0$, with respective domains $\mathcal{D} = \{D_1, D_2, \ldots, D_k\}$. A *constraint* C on $\mathcal{V}$ is defined as a subset of the Cartesian product of the domains of the variables in $\mathcal{V}$, i.e. $C \subseteq D_1 \times D_2 \times \ldots \times D_k$. The cardinality of $\mathcal{V}$, $|\mathcal{V}|$, is the *arity* of C. C is a *unary constraint* if it has arity 1, it is a *binary constraint* if it has arity 2, and it is a *non-binary constraint* if it has arity k, with $k > 2$. Finally, C is a *global constraint* if it is a relation among a non-fixed number of variables.

A *Constraint Satisfaction Problem* (CSP) [1, 17, 62] is a triple $\langle \mathcal{V}, \mathcal{C}, \mathcal{D} \rangle$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_k\}$ is a finite set of variables with respective domains $\mathcal{D} = \{D_1, D_2, \ldots, D_k\}$, and $\mathcal{C}$ is a finite set of constraints, each of which is defined on a subset of the variables in $\mathcal{V}$.

Consider a CSP $\langle \mathcal{V}, \mathcal{C}, \mathcal{D} \rangle$, a tuple $(d_1, \ldots, d_k) \in D_1 \times D_2 \times \ldots \times D_k$ *satisfies* a constraint $C_i \in \mathcal{C}$ on the variables $v_{i1}, v_{i2}, \ldots, v_{im}$ if $(d_{i1}, d_{i2}, \ldots, d_{im}) \in C_i$. A tuple $(d_1, \ldots, d_k) \in D_1 \times D_2 \times \ldots \times D_k$ is a *solution* to a CSP if it satisfies every constraint $C \in \mathcal{C}$.

Consider the CSPs $P = \langle \mathcal{V}, \mathcal{C}, \mathcal{D} \rangle$ and $P' = \langle \mathcal{V}, \mathcal{C}', \mathcal{D}' \rangle$. P and P' are called *equivalent* if they have the same solution set. P is said to be *smaller* that P' if they are equivalent and $D_i \subseteq D'_i$ for all i. This relation is written as $P \preceq P'$. P is *strictly smaller* that P', if $P \preceq P'$ and $D_i \subset D'_i$ for at least one i. This is written $P \prec P'$. When both $P \preceq P'$ and $P' \preceq P$ we write $P \equiv P'$

Often we want to find a solution to a CSP that is optimal with respect to certain criteria. Consider a CSP $\langle \mathcal{V}, \mathcal{C}, \mathcal{D} \rangle$, where $\mathcal{D} = \{D_1, D_2, \ldots, D_k\}$. Let $\mathcal{S}$ be the *solution set*, that is the set of all the tuples $(d_1, \ldots, d_k) \in D_1 \times D_2 \times \ldots \times D_k$ that are solutions to the CSP. A *Constraint Optimization Problem*, or a COP, is a CSP on the solution set of which an *objective function*, $f : \mathcal{S} \rightarrow \mathbb{R}$, has to be optimized. An *optimal solution* to a COP is a solution to the CSP that is optimal with respect to f. The objective function value is often represented by a variable z, together with the "constraint" `maximize` z or `minimize` z, respectively for a maximization or a minimization problem.

In Constraint Programming (CP), the goal is to find a solution (or all solutions) to a given CSP, or an optimal solution (or all optimal solutions) to a given COP. A filtering algorithm is typically associated with every constraint. This algorithm re-

moves values from the domains of the variables participating in the constraint that cannot belong to any solution of the CSP. These filtering algorithms are repeatedly called until no new deduction can be made. This process is called *constraint propagation* or *propagation* in short. In conjunction with this process CP uses a search procedure (like a backtracking algorithm) where filtering algorithms are systematically applied when the domain of a variable is modified. The solution process interleaves *propagation* and *search* to reach the given goal.

Example 1.2.1. Let x_1, x_2 be variables with respective domains $D_1 = \{0, 1, 2, 3\}$, $D_2 = \{0, 1, 2, 3, 4, 5\}$. Let x_3 be a binary variable with domain $D_3 = \{0, 1\}$. On these variables we impose the following constraints: $x_1 \geq 3$ $x_1 + x_2 \geq 8$ and $(x_2 > 0) \leftrightarrow (x_3 = 1)$. We denote the resulting CSP as

$$
\begin{aligned}
& x_1 \in \{0, 1, 2, 3\}, x_2 \in \{0, 1, 2, 3, 4, 5\}, x_3 \in \{0, 1\}, \\
& x_1 \geq 3, \\
& x_1 + x_2 = 8, \\
& (x_2 > 0) \leftrightarrow (x_3 = 1).
\end{aligned}
$$

A solution to this CSP is $x_1 = 3$, $x_2 = 5$ and $x_3 = 1$. ◇

Propagation. Constraint propagation is a process that removes a subset or all the inconsistent values from the domains, by reasoning on the individual constraints. This process may significantly reduce the search space. Thus constraint propagation is a key instrument to improve the efficiency of CP solvers.

Let C be a constraint on the variables $x_1, \ldots, x_m$ with respective domains $D_1, \ldots, D_m$. A *propagation algorithm* for C removes values from $D_1, \ldots, D_m$ that do not participate in a solution to C. A propagation algorithm does not have to remove *all* such values, as this may lead to an exponential running time due to the nature of some constraints.

We consider the CSP $P = \langle \mathcal{V}, \mathcal{C}, \mathcal{D} \rangle$. P can be transformed into a smaller CSP P' by repeatedly applying the propagation algorithm for all constraints in $\mathcal{C}$ until there is no more domain reduction. This process is called *constraint propagation*. When no more domain reduction can be achieved by iterating the process, we say

that each constraint, and the CSP, is *locally consistent* and that we have achieved a notion of *local consistency* on the constraints and the CSP. The term "local consistency" reflects the fact that the CSP obtained through the discussed process is not globally consistent. It is instead a CSP in which all the constraints are "locally", i.e. individually, consistent. A comprehensive discussion on the process of constraint propagation is given by Apt [1].

If we demand that every domain value of every variable in the constraint belongs to a solution to the constraint then what we achieve is *hyper-arc consistency*, that is the strongest local consistency notion for a constraint. This still does not guarantee a solution to the whole CSP because other constraints in it are not considered in such a process.

Example 1.2.2. Consider again the CSP of Example 1.2.1, i.e. variables x_1, x_2, x_3 with respective domains $D_1 = \{0, 1, 2, 3\}$, $D_2 = \{0, 1, 2, 3, 4, 5\}$, $D_3 = \{0, 1\}$, and

$$x_1 \geq 3, \tag{1.1}$$

$$x_1 + x_2 = 8, \tag{1.2}$$

$$(x_2 > 0) \leftrightarrow (x_3 = 1). \tag{1.3}$$

We apply constraint propagation until the constraints are hyper-arc consistent:

$$\begin{array}{l} x_1 \in \{0,1,2,3\} \\ x_2 \in \{0,1,2,3,4,5\} \\ x_3 \in \{0,1\} \end{array} \overset{(1.1)}{\longrightarrow} \begin{array}{l} x_1 \in \{3\} \\ x_2 \in \{0,1,2,3,4,5\} \\ x_3 \in \{0,1\} \end{array} \overset{(1.2)}{\longrightarrow} \begin{array}{l} x_1 \in \{3\} \\ x_2 \in \{5\} \\ x_3 \in \{0,1\} \end{array} \overset{(1.3)}{\longrightarrow} \begin{array}{l} x_1 \in \{3\} \\ x_2 \in \{5\} \\ x_3 \in \{1\} \end{array}$$

◇

The three constraints are examined sequentially, as indicated above the arcs. We first examine constraint 1.1, and deduce that values 0,1 and 2 in D_1 do not appear in a solution to it. Then we examine constraint 1.2, and remove all the values except 5 from D_2. This is because 5 is the only value that supports the remaining value 3 in D_1. Finally we examine constraint 1.3 and we remove value 0 from D_3. The resulting CSP is hyper-arc consistent. In fact, we found a solution to the CSP.

The method applied to make a CSP locally consistent should be as efficient

as possible, in fact constraint propagation is applied each and every time a decision variable domain has been changed. This happens very frequently during the solution process. Note that both the efficiency of the propagation algorithms and the order in which the propagation algorithms are applied directly influences the efficiency of constraint propagation.

Search. In the solution process CP employs a *search tree*. A search tree is composed by a set of vertices, or *nodes*, and a set of arcs, or *branches*. A node v is a *direct descendant* of a node u and, conversely, u is the *parent* of v, if (u, v) is an arc of a search tree.

Definition 1.2.1 (Search tree [1])**.** *Let P be a CSP with a sequence of variables X. A search tree for P is a (finite) tree such that*

- *its nodes are CSPs,*
- *its root is P,*
- *if $P_1, \ldots, P_m$ where $m > 0$ are all direct descendants of P_0, then the union of $P_1, \ldots, P_m$ is equivalent w.r.t. X to P_0, for every node P_0.*

A node P of a search tree is at *depth* d if the length of the path from the root to P is d.

In CP, a search tree is dynamically built by splitting a CSP into smaller CSPs, until we reach an inconsistency , i.e. some decision variable domain becomes empty, or a solution to the CSP. There are two possible ways to split a CSP into smaller CSPs: we can either split a constraint (for instance a disjunction) or split the domain of a variable. The second being the most common technique.

A direct consequence of what we discussed is that a CSP is associated with each node in the search tree. At each node we can therefore apply constraint propagation and we may detect that the corresponding CSP is inconsistent, or we may achieve some domain reduction for it. Obviously, in both cases we will generate and explore less nodes, this is the reason why constraint propagation can speed up the solution process. However, in order to do so, constraint propagation must be efficient. This means that the time spent on propagation should be less than the time that is gained by it.

In splitting the domain of a variable, we first select a variable and then decide how to split its domain. This process is guided by **variable** and **value ordering heuristics**. These heuristics impose an ordering on the variables and values, respectively. The ordering imposed by these heuristics has a great impact on the search process.

First we give the following definitions that are relevant to introduce variable and value ordering heuristics. A relation $\preceq$ on a set S is called a *partial order* if it is reflexive ($s \preceq s$ for all $s \in S$), transitive ($s \preceq t$ and $t \preceq u$ implies $s \preceq u$), and antisymmetric ($s \preceq t$ and $t \preceq s$ implies $s = t$). A partial order $\preceq$ is a *total order* if $s \preceq t$ or $t \preceq s$ for all $t, s \in S$. Given a partial order $\preceq$ on a set s, an element $s \in S$ is called a *least* element if $s \preceq t$ for all $t \in S$. Two elements $s, t \in S$ are *incomparable* with respect to $\preceq$ if $s \npreceq t$ and $t \npreceq s$.

A *variable ordering heuristic* imposes a partial order on the variables with non-singleton domains. The *most constrained first* variable ordering heuristic is of common use. Variables are ordered according to the respective number of occurrences in the constraints. A variable that appears the most often, is ordered least. The ratio behind this is that, most likely, changing the domains of such variables will cause more values to be removed by constraint propagation. Another variable ordering heuristic is the *smallest domain first* heuristic, also known as the *first fail* heuristic. Variables are ordered, in this heuristic, with respect to the size of their domains. A variable that has the smallest domain is ordered least. By using this heuristic less nodes are generated in the search tree and inconsistent CSPs are detected earlier. If two or more variables are incomparable, a common strategy is to apply the lexicographic ordering to the variables in order to obtain a total order.

A *value ordering heuristic* induces a partial order on the domain of a variables. Values in the domains are ordered according to a certain criterion, such that values that are ordered least are selected first. For instance, the *lexicographic* value ordering heuristic orders the values according to a lexicographic ordering. The *random* value ordering heuristic, instead, orders the variables randomly. Similarly to what discussed for the variable ordering heuristics, if a value ordering heuristic imposes a partial order on a domain, we can apply the lexicographic or random value ordering heuristic to incomparable values in order to create a total

order. A value ordering heuristic is also referred to as *branching heuristic* because it decides the order of the branches in the search tree.

A **domain splitting procedure** is applied after a variable has been selected and a value ordering heuristics imposed a total order on its domain. Given a domain, a *domain splitting procedure* generates a partition of the domain. Consider a domain $D = \{d_1, d_2, \ldots, d_m\}$ and a total order $\preceq$ such that $d_1 \preceq d_2 \preceq \ldots \preceq d_m$. Two common domain splitting procedures are labeling and bisection. *Labeling* splits D into $\{d1\}, \{d_2\}, \ldots, \{d_m\}$. In practice the labeling procedure is often implemented to split a domain D into $\{d1\}, \{d_2, \ldots, d_m\}$. This procedure is also called *enumeration* in the literature. *Bisection* splits D into $\{d1, \ldots, d_k\}, \{d_{k+1}, \ldots, d_m\}$, where $k = \lfloor m/2 \rfloor$.

Consider a CSP $P_0 = \langle \mathcal{V}, \mathcal{C}, \mathcal{D} \rangle$ and a variable $v \in \mathcal{V}$ whose domain has been split into the partition $D_1, \ldots, D_k$. Then we define the direct descendants of P_0 as $P_i = \langle \mathcal{V}, \mathcal{C} \cup \{v \in D_i\}, \mathcal{D} \rangle$ for $i = 1, \ldots, k$. In practice, we modify the domain of a variable instead of adding a constraint to define a descendant. If the partition "respects" the value ordering heuristic that was applied to the domain, i.e. $d_i \preceq d_j$ for all $d_i \in D_i$, $d_j \in D_j$, $i < j$ and $i = 1, \ldots, k-1$, the corresponding descendants inherit the ordering of the value ordering heuristic, i.e. $P_1 \preceq \ldots \preceq P_k$.

A **search strategy** defines the *traversal* of the search tree. Assume that all the direct descendants of a node in a search tree are totally ordered, for instance according to the given value ordering heuristic. The least element corresponds to the first descendant.

Depth-first search (DFS): starting from the root node, proceed by descending to its first descendant. Continue until a leaf is reached, then backtrack to the parent of the leaf and descend to its next descendant, if it exists. Continue the process until the root node is reached again and all its descendants have been visited. DFS is a complete (or exact) search strategy, not redundant. This means that it explores all paths from the root to a leaf exactly once. In DFS backtracking to a previous node only takes place after we have visited a leaf. This leads to the more general notion of depth-first based search strategies.

Depth-first based search strategies: we start at the root node and we proceed by descending to its first descendant. This process continues until a leaf

is reached. Then we backtrack to some previously visited node and descend to its next descendant, if it exists and if it is allowed. This process continues until all leafs have been visited. Other examples of depth-first based search strategies, in addition to DFS, are limited discrepancy search or LDF [38], depth-bounded discrepancy search or DDS [97], and discrepancy-bounded depth-first search, or DBDFS [7].

Optimization. By recalling that a COP consists of a CSP together with an objective function f, it is easy to see why the search for an optimal solution (or all the optimal solutions) to a COP operates in a similar fashion to the search for a solution to a CSP. By restricting (without loss of generality) ourselves to minimization problems, we represent the objective value using a variable z. When a solutions to the CSP is found, the corresponding value of z, say $z = \beta$, represents an upper bound for the optimal value of f. It follows that we can add the constraint $z < \beta$ to all the CSPs in the search tree and continue. This will, in practice, replace the maximum value in the domain of z with β.

Example 1.2.3. We present the solution process of CP, using the following COP P_0:

$$
\begin{aligned}
& x_1 \in \{3, 8\}, x_2 \in \{0, 1, 2, 3, 4, 5\}, x_3 \in \{0, 1\}, \\
& \texttt{minimize}\ z, \\
& z = x_1 + 6x_3, \\
& x_1 \geq 3, \\
& x_1 + x_2 = 8, \\
& (x_2 > 0) \leftrightarrow (x_3 = 1).
\end{aligned}
$$

To build a search tree, we apply the lexicographic variable and value ordering heuristic and use labeling as domain splitting procedure. As search strategy we use DFS. At each node we apply hyper-arc consistency constraint propagation. The CSP P_0 is the root. The search tree is depicted in Fig. 1.1. We first apply

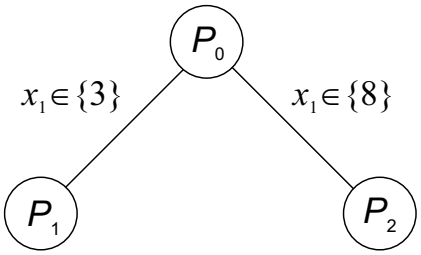


Figure 1.1: The search tree of Example 1.2.3

constraint propagation to P_0. It follows that

$$x_1 \in \{3, 8\}, x_2 \in \{0, 5\}, x_3 \in \{0, 1\}, z \in \{8, 9\}.$$

We select the lexicographically least variable, x_1, split its domains into $\{3\}$ and $\{8\}$, and generate the descendants P_1 and P_2 where $P_1 = P_0 \cup x_1 \in \{3\}$ and $P_2 = P_0 \cup x_1 \in \{8\}$.

We descend to node P_1 and apply constraint propagation. It follows that

$$x_1 \in \{3\}, x_2 \in \{5\}, x_3 \in \{1\}, z \in \{9\}.$$

We have found a solution with $z = 9$. Hence we add to all CSPs the constraint $z < 9$.

Next we backtrack to P_0, descend to P_2, and apply constraint propagation. It follows that

$$x_1 \in \{8\}, x_2 \in \{0\}, x_3 \in \{0\}, z \in \{8\}.$$

We have found a solution with $z = 8$. Hence we add to all CSPs the constraint $z < 8$. Next we backtrack to P_0 and stop because all its descendants have been visited.

We return the optimal solution we found in leaf P_2. ◇

Optimization-oriented global constraints embed an optimization component, representing a proper relaxation of the constraint itself, into a global constraint [32]. The relaxation employed can be a continuous relaxation, as in the examples provided in [32], a DP relaxation, as discussed in [33], or it can be

any other suitable relaxation. The optimization component provides three pieces of information: (a) the optimal solution of the relaxed problem; (b) the optimal value of this solution representing an upper bound on the original problem objective function; (c) a *gradient function* **grad(**V**,**v**)**, which returns for each couple variable-value (V,v) an optimistic evaluation of the profit obtained if v is assigned to V. These pieces of information are exploited both for propagation purposes and for guiding the search.

1.2.3 Stochastic Constraint Programming

In order to extend CP to decision problems under uncertainty, Walsh [98] proposed an extension of CP called Stochastic Constraint Programming (SCP) in which there is a distinction between decision variables, which can be set freely, and stochastic (or observed) variables, which follow some probability distribution.

We first provide some basic notions on probability theory.

Probability theory. In probability theory uncertainty is represented in terms of random experiments. Let ω be an outcome of an experiment, the set of all the possible outcomes is represented by Ω.

Subsets of Ω are called *events*, which combine one or more outcomes. We denote by $\mathcal{A}$ a collection of random events.

To each event $A \in \mathcal{A}$ is associated a value $\Pr\{A\}$, called a *probability*, such that $0 \leq \Pr\{A\} \leq 1$, $\Pr\{\emptyset\} = 0$, $Pr\{\Omega\} = 1$ and $\Pr\{A_1 \cup A_2\} = \Pr\{A_1\} + \Pr\{A_2\}$ if $A_1 \cap A_2 = \emptyset$.

The triplet $\langle \Omega, \mathcal{A}, \Pr \rangle$ is called a *probability space* that must satisfy a number of conditions (see, e.g, [54]). Several random variables associated with a probability space can be defined, namely, all the variables that are influenced by the random events in $\mathcal{A}$.

In some cases the elements $\omega \in \Omega$ are used to describe a few *states of the world* or *scenarios*. All random elements then jointly depend on these finitely many scenarios.

For a particular random variable ξ, its cumulative distribution is defined as $F_\xi(x) = \Pr\{\xi \leq x\}$, or more precisely $F_\xi(x) = \Pr\{\{\omega | \xi \leq x\}\}$.

A discrete random variable takes a finite or countable number of different values. It is described by its probability mass function. This is the list of possible values ξ_k, $k \in K$, with associated probability

$$f_\xi(\xi_k) = \Pr\{\xi = \xi_k\},$$

such that $\sum_{k \in K} f_\xi(\xi_k) = 1$.

Continuous random variables can often be described through a so-called *density* function $f_\xi(\xi)$. The probability of ξ being in an interval $[a, b]$ is obtained as

$$\Pr\{a \leq \xi \leq b\} = \int_a^b f_\xi(\xi) \mathrm{d}\xi,$$

or equivalently

$$\Pr\{a \leq \xi \leq b\} = \int_a^b \mathrm{d}F_\xi(\xi),$$

where $F_\xi(.)$ is the cumulative distribution as earlier. Contrary to the discrete case, the probability of a single value $\Pr\{\xi = a\}$ is always zero for a continuous random variable. The distribution $F_\xi(.)$ must be such that $\int_{-\infty}^{\infty} \mathrm{d}F_\xi(\xi) = 1$.

The *expectation* of a random variable is computed as $\mu = \sum_{k \in K} \xi_k f_\xi(\xi_k)$ or $\mu = \int_{-\infty}^{\infty} \xi \mathrm{d}F_\xi(\xi)$ in the discrete and in the continuous case, respectively.

The *variance* of a random variable is $\mathrm{E}[(\xi - \mu)^2]$, where $\mathrm{E}[.]$ denotes the expectation.

The expectation of ξ^r is called the *r-th moment* of ξ. A point η is called the *α-quantile* of ξ if and only if for $0 < \alpha < 1$, $\eta = \min\{x | F_\xi(x) \geq \alpha\}$.

Let $\phi : \mathbb{R} \to \mathbb{R}$ be a convex function and ξ a random variable. Then $\phi(\mathrm{E}[\xi]) \leq \mathrm{E}[\phi(\xi)]$ (*Jensen's inequality*).

Equipped with these notions, we now formally introduce SCP.

Semantics. A *stochastic CSP* is defined as a 6-tuple $\langle V, S, D, P, C, \theta \rangle$, where V is a set of decision variables and S is a set of stochastic variables, D is a function mapping each element of V and each element of S to a domain of potential values. A decision variable in V is *assigned* a value from its domain. P is a function mapping each element of S to a probability distribution for its associated domain. C is a set of constraints. A constraint $h \in C$ that constrains at least one variable

in S is a *chance-constraint*. θ_h is a threshold value in the interval $[0, 1]$, indicating the minimum satisfaction probability for chance-constraint h. Note that a chance-constraint with a threshold of 1 (or without any explicit threshold specified) is equivalent to a hard constraint.

A stochastic CSP consists of a number of *decision stages*. A decision stage is a pair $\langle V_i, S_i \rangle$, where V_i is a set of decision variables and S_i is a set of stochastic variables.

One-stage Stochastic CSP. In a one-stage stochastic CSP, a single stage is considered, $\langle V, S \rangle$, and the decision variables are set before observing the realizations of the stochastic variables. A solution can be therefore expressed as an assignment for decision variables in V such that, given random values for stochastic variables in S, the hard constraints are satisfied and the chance-constraints are satisfied in the specified fraction of all possible scenarios.

m-stage Stochastic CSP. In an m-stage stochastic CSP, V and S are partitioned into disjoint sets, $V_1, \ldots, V_m$ and $S_1, \ldots, S_m$, and we consider multiple stages, $\langle V_1, S_1 \rangle, \langle V_2, S_2 \rangle, \ldots, \langle V_m, S_m \rangle$. A decision variable $x_i \in V_j$ is set to a value only after realizations of stochastic variables $\left\{ y_i \,\middle|\, y_i \in \bigcup_{t=1}^{j-1} S_t \right\}$ in former stages have been observed. To solve an m-stage stochastic CSP an assignment to the variables in V_1 must be found such that, given random values for S_1, assignments can be found for V_2 such that, given random values for S_2, ..., assignments can be found for V_m so that, given random values for S_m, the hard constraints are satisfied and the chance constraints are satisfied in the specified fraction of all possible scenarios. The solution of an m-stage stochastic CSP is represented by means of a *policy tree* [91]. A policy tree is a set of decisions where each path represents a different possible scenario and the values assigned to decision variables in this scenarios.

Stochastic Constraint Optimization. Let $\mathcal{S}$ denote the space of policy trees representing all the solutions of a stochastic CSP. We may be interested in finding a feasible solution, i.e. a policy tree $s \in \mathcal{S}$, that maximizes the value of a given objective function $f(\cdot)$ over the stochastic variables S (edges of the policy tree)

and over a subset $\widehat{V} \subseteq V$ of the decision variables (nodes in the policy tree). A *Stochastic COP* is then defined in general as

$$\max_{s \in \mathcal{S}} f(s).$$

Solution methods. Two solution methods have been proposed so far in the literature: the first relies on backtracking and forward checking algorithms proposed in [98], the second [91] adopts a scenario based approach.

Policy based view. In the policy based view of [98], the semantics is based on a tree of decisions. Each path in a policy represents a different possible scenario (set of values for the stochastic variables), and the values assigned to decision variables in this scenario. To find satisfying policies, backtracking and forward checking algorithms, which explores the implicit AND/OR graph, are presented. Stochastic variables give AND nodes as we must find a policy that satisfies all their values, whilst decision variables give OR nodes as we only need find one satisfying value. In [5] the authors extend the forward checking procedure to better take advantage of probabilities and thus achieve stronger pruning. They also define arc consistency for stochastic CSPs and introduce an arc consistency algorithm that can handle constraints of any arity.

Scenario based approach. In a scenario based approach [11, 91], a scenario tree is generated which incorporates all possible realization of discrete random variables into the model explicitly. A path from the root to an extremity of the event tree represents a scenario $\omega \in \Omega$, where Ω is the set of all possible scenarios. With each scenario a given probability is associated. If S_i is the ith random variable on a path from the root to the leaf representing scenario ω and a_i is the value given to S_i in the ith stage of this scenario, then the probability of this scenario is given by $\Pr\{\omega\} = \prod_i \Pr(S_i = a_i)$. Within each scenario, we have a conventional (non-stochastic) constraint program to solve. All we have to do is replacing the stochastic variables by the values taken in the scenario and ensure that the values found for the decision variables are consistent across scenarios as certain decision variables are shared across scenarios. Constraints are defined (as in tradi-

tional constraint satisfaction) by relations of allowed tuples of values, and can be implemented with specialized and efficient algorithms for consistency checking. The great advantage of this approach is that conventional constraint solvers can be used to solve stochastic constraint programs. The scenario-based view of stochastic constraint programs also allows later stage stochastic variables to take values which are conditioned by the earlier stage stochastic variables. This is a direct consequence of employing the scenario representation, in which stochastic variables are replaced with their scenario dependent values. Of course, there is a price to pay as the number of scenarios grows exponentially with the number of stages. For this reason, the authors also proposed several approximate solution methods based on scenario reduction methods. We here mention two of the reduction approaches employed. The "most likely scenario" approach only consider a few of the most probable scenarios and ignore rare events. Latin Hypercube Sampling [84] ensures that the ensemble of random numbers is representative of the real variability whereas traditional random sampling (sometimes called brute force) is just an ensemble of random numbers without any guarantee.

1.2.4 Inventory Control

In the previous sections we formally introduced CP and its extension for decision problems under uncertainty, SCP. We now introduce the relevant formal background in inventory control, since the rest of this dissertation will extensively discuss the application of SCP techniques to inventory control problems.

This section is mainly based on [81].

Lot-sizing is a very active research area in combinatorial optimization. Analyzing and controlling inventory systems that have to cope with dynamic demand patterns is a challenging task [14, 30]. Therefore it does not surprise that controlling stochastic inventory systems is even harder and that stochastic multi-period lot-sizing problems currently represent a challenging research area [100]. In the following sections we will provide some background on stochastic lot-sizing.

In lot-sizing problems, when the demand is assumed to be stochastic, the cost of insufficient capacity in the short run — that is the cost associated with shortages, or with averting them — assumes a great importance. The problem in

stochastic lot-sizing is typically to determine the "correct" quantity of buffer (or safety) stocks that must be kept to meet unexpected fluctuations of the demand. There are several possible choices for the shortage costing method or for the customer service level measure. In what follows we will firstly recall the Newsvendor problem, probably the most studied problem in stochastic lot-sizing, and then we will extend the discussion to multi-period stochastic lot-sizing problems under continuous and periodic review strategies.

The Newsvendor problem. The Newsvendor problem is the prototype of the problem faced by a news vendor who needs to decide how many newspapers to buy and stock on a news stand before observing demand. In other words, it is the problem of controlling the inventory of a single item with stochastic demand over a single period. As demand occurs, he may face both overage costs — if he orders too much — or underage costs — if he orders too little. Therefore he must hedge against overage costs and underage costs in order to minimize the respective effects. The problem becomes particularly significant for problem with high demand uncertainty and large overage and underage costs.

The problem inputs are as follows:

- d: the one period random demand, with mean $\mu = \mathrm{E}[d]$ and variance $\sigma^2 = V[d]$
- c: the unit cost,
- p: the selling price, where $p > c$
- s: the salvage value, where $s < c$.

If x units are ordered, then $\min(x, d)$ units are sold and $(x - D)^+ = \max(x - d, 0)$ units are salvaged.

The news vendor profit is given by $p\min(x, d) + s(x-d)^+ - cx$. The expected profit is well defined and given by:

$$\pi(x) = p\mathrm{E}[\min(x, d)] + s\mathrm{E}[(x - d)^+] - cx.$$

We now use the fact that $\min(x, d) = d - (d - x)^+$ and we rewrite the expected profit as

$$\pi(x) = (p - c)\mu - G(x) \tag{1.4}$$

where

$$G(x) = (c - s)\mathrm{E}[x - d)^+ + (p - c)\mathrm{E}[(d - x)^+] \geq 0.$$

Often, it is convenient to formulate the problem in terms of per unit holding and penalty cost. Let $h = c - s$ and $b = p - c$, where h and b are respectively the per unit holding cost and penalty cost. Sometimes the penalty cost is inflated to take into account the *ill-will* cost associated with unsatisfied demand. Eq. 1.4 allows us to view the problem of maximizing $\pi(x)$ as that of minimizing the expected holding and penalty cost $G(x)$.

Obviously, the Newsvendor problem is only interesting when the demand is random. In fact, let $G^{det}(x) = h(\mu - x)^+ + b(x - \mu)^+$ be the cost when d is deterministic, i.e. $\Pr\{d = \mu\} = 1$. Then $x = \mu$ minimizes $G^{det}(x)$ and $G^{det}(\mu) = 0$, so $\pi^{et}(\mu) = (p - c)\mu$. The problem is also trivial when $s = c$. In this (unrealistic) case we can order an infinite amount, satisfy all the demand, and then return all the unsold items.

We now introduce $g(x) = hx^+ + bx^-$. $G(x)$ can then be rewritten as $G(x) = \mathrm{E}[g(x - d)]$. Function g is convex, by recalling that convexity is preserved by linear transformations and by expectation operator, it follows that G is also convex. By Jensen's inequality [54] $G(x) \geq G^{det}(x)$. As a result, $\pi(x) \leq \pi^{det}(x) \leq \pi^{det}(\mu) = (p - c)\mu$. We can never expect, in the stochastic case, a higher profit than the one obtained when the demand is deterministic. Note that this result immediately suggests an effective strategy for obtaining bounds for convex stochastic programs, in Chapter 4 we will present in details such a strategy.

Consider a continuous distribution for d, an optimal solution to the above problem can be found by taking the derivative of G and setting it to zero. Since we can interchange the derivative and the expectation operators, it follows that $G'(x) = h\mathrm{E}[\delta(x - d) - b\mathrm{E}[\delta(d - x)$ where $\delta(x) = 1$ if $x > 0$ and zero otherwise. Since $\mathrm{E}[\delta(x - d)] = \Pr\{x - d > 0\}$ and $\mathrm{E}[\delta(d - x)] = \Pr\{d - x > 0\}$, it follows that

$$G'(x) = h\Pr\{x - d > 0\} - b\Pr\{d - x > 0\}.$$

Setting the derivative to zero reveals that

$$F(x) \equiv \Pr\{d \leq x\} = \frac{b}{b+h} = \frac{p-c}{p-s} \equiv \alpha. \tag{1.5}$$

when F is continuous then at least one x exists that satisfies Eq. 1.5. We select the smallest of these by letting

$$x^* = \inf\{x \geq 0 | F(x) \geq \alpha\}.$$

Clearly x^*, selected this way, is increasing in α and therefore it is increasing in b and decreasing in h.

When F is strictly increasing then the inverse function F^{-1} exists and there is a unique optimal solution given by

$$x^* = F^{-1}(\alpha). \tag{1.6}$$

Nevertheless, d is often defined over the set of natural numbers $\mathbb{N} = \{0, 1, \ldots\}$. In this case we must consider the forward difference $\Delta G(x) = G(x+1) - G(x)$, $x \in \mathbb{N}$. By writing $\mathrm{E}[(d-x)^+] = \sum_{j=x}^{\infty} \Pr\{d > j\}$, it is easy to see that

$$\Delta G(x) = h - (h+b) \Pr\{d > x\}$$

is non-decreasing in x, and that $\lim_{x\to\infty} \Delta G(x) = h > 0$, so an optimal solution is given by $x^* = \min\{x \in \mathbb{N} | \Delta G(x) \geq 0\}$, or equivalently,

$$x^* = \min\{x \in \mathbb{N} | F(x) \geq \alpha\},$$

The Newsvendor model dates back to the 1888 paper by Edgeworth [26] who used the Central Limit Theorem to determine the amount of cash to keep at a bank in order to satisfy random cash withdrawals from deposit with high probability. The fractile solution 1.5 appeared in 1951 in Arrow, Harris and Marchar [2].

The Newsvendor solution can be interpreted as the smallest order quantity that guarantees that all demand will be satisfied with probability $100\alpha\%$. In practice, managers often specify α and then find x^* accordingly. This service

level, also known as *cycle service level*, should not be confused with the fill-rate, that is instead the fraction of demand served from stock. This is defined as $\beta = \mathrm{E}[\min(d, x)]/\mathrm{E}[d]$.

Normal demand distribution. Particularly interesting is the case when the demand d is normally distributed. This assumption can be often justified by the Central Limit Theorem, when the demand comes from many different independent or weakly dependent customers.

If d is normally distributed, then $d = \mu + Z\sigma$, where Z is a standard normal random variable (i.e. a normally distributed random variable with mean 0 and variance 1). Let $\Phi(z) = \Pr\{Z < z\}$ be the cumulative distribution function of the standard normal random variable. Although the function Φ is not available in closed form, it is available in tables [52].

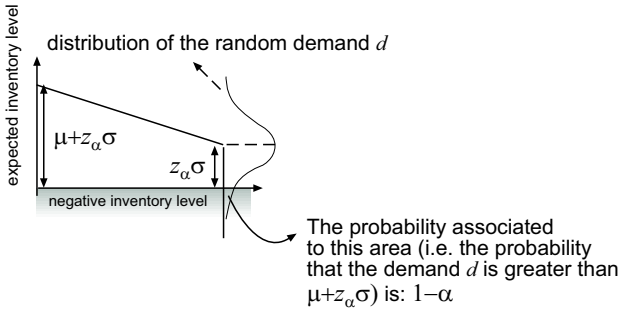


Figure 1.2: Newsvendor problem with normally distributed demand

Since $\Pr\{d \leq \mu + z_\alpha\sigma\} = \Phi(z_\alpha) = \alpha$, it follows that

$$x^* = \mu + z_\alpha\sigma$$

satisfies Eq. 1.6, so it gives the optimal solution for the case of normal demand. The quantity z_α is known as *safety factor* and $x^* - \mu = z_\alpha * \sigma$ is known as the *safety stock*, Fig. 1.2.

It can be shown that

$$G(x^*) = (h + b)\sigma\phi(z_\alpha),$$

where ϕ is the density function of the standard normal random variable, and that

$$\pi(x^*) = (p-c)\mu - (p-s)\sigma\phi(z_\alpha).$$

In addition the fill-rate can be also easily written as

$$\beta = 1 - c_v[\phi(z_\alpha) - (1-\alpha)z_\alpha]$$

where $c_v = \sigma/\mu$ is the coefficient of variation of the demand. Since $\phi(z_\alpha) - (1-\alpha)z_\alpha \geq 0$ is decreasing in α, it follows that β is increasing in α and decreasing in c_v. Notice, for example that $\beta = 0.97$ when $\alpha = 0.75$ and $c_v = 0.2$, while $\beta = 0.991$ when $\alpha = 0.9$ and $c_v = 0.2$.

Example 1.2.4. Suppose that d is normal with mean $\mu = 100$ and standard deviation $\sigma = 20$. If $c = 5$, $h = 1$ and $b = 3$, then $\alpha = b/(b+h) = 0.75$ and $x^* = 100 + 0.6745 \cdot 20 = 113.49$, in fact $\Phi^{-1}(0.75) \cong 0.6745$. Note that the order is for 13.49 units (safety stock) more than the mean. Note also that $\phi(0.6745) = 0.3178$ so $G(113.49) = 4 \cdot 20 \cdot 0.3178 = 25.42$ and $\pi(113.49) = 274.58$, with $\beta = 0.97$. $\diamond$

Inventory control policies. In the previous paragraph we introduced the Newsvendor problem. The key aspect of this problem is the fact that a *single* replenishment decision concerning an order quantity has to be taken in advance, to meet the random demand till the end of the time horizon considered.

Nevertheless, what usually happens in the reality is that management has to take multiple decisions to meet the demand. These decisions usually concern the number of planned replenishments, the timing of such replenishments, and the quantity of items that has to be ordered at each replenishment. Obviously there are many different *strategies* to decide on replenishment periods and replenishment quantities. For instance we could fix a rule stating that a replenishment should be performed every time the inventory level falls below a given threshold. In this case the decision would concern two aspects: choosing the "threshold" and the quantity that has to be ordered when the inventory position falls below this threshold. Alternatively, a strategy could consist in ordering according to prede-

fined time intervals. Moreover, instead of deciding in advance the exact quantity to be ordered, we could instead try to fix a level for each replenishment (order-up-to-level) up to which we will raise the stocks. Each of these different strategies constitutes an *inventory control policy*.

For many replenishment policies a challenging problem is that of finding the optimal "settings", for instance the reorder levels and the order-up-to-levels minimizing some cost structure or meeting certain service level requirements [89]. Often people are also interested in comparing different policies in such a way to determine which policy always guarantees the best cost performance [76].

Notation and terminology. We shall now introduce some important issues and terminology concerning inventory control policies. When demand is stochastic, it is useful to conceptually categorize inventories as follows:

- *On-hand stock:* This is stock that is physically on the shelf; it can never be negative. This quantity is relevant in determining whether a particular customer demand is satisfied directly from the shelf;

- *Backorders:* These denote an existing demand that cannot be fulfilled since no stock is available on the shelf;

- *On order:* These are stocks which have been ordered, but that for some reason have not reached the shelf yet. Reasons for this may comprise: stock inspection, transportation etc.;

- *Net stock* = (On hand) - (Backorders).

  This quantity can become negative (namely, if there are backorders). It is used in some mathematical derivations and is also a component of the following important definition:

- *Inventory position:* The inventory position is defined by the relation

  Inventory position = (On hand) + (On order) - (Backorders).

As we will see, the inventory position is a key quantity for replenishment decisions.

- *Safety stock:* The safety (or buffer) stock is defined as the average level of the net stock just before a replenishment arrives. If the safety stock is zero, this means that, on average, we will run out of stock at the moment when a replenishment arrives. A positive safety stock provides a buffer to hedge against larger-than-average demand between subsequent replenishment arrivals. The numerical value of the safety stock depends, as we will see, on what happens to demands when there is a stockout.

What happens to a customer's order when an item is temporarily out of stock is of obvious importance in inventory control. There are two extreme cases.

- *Complete backordering:* When a stockout occurs, demands are backordered and filled as soon as an adequate-sized replenishment arrives.

- *Complete lost sales:* When a stockout occurs, demands are lost until a replenishment arrives; customers go elsewhere to satisfy their needs.

Although most inventory models have been developed for one or the other of these two extreme situations, in many practical situation we find a combination of these two extremes. The ratio behind the inventory models commonly in use is that the decisions they produce tend to be quite insensitive to the degree of backordering possible in particular situation. The reason for this is that in practice high customer service levels are used, which implies infrequent stockout occasions. When we use the term *stockout*, we mean a stockout occasion or event. The number of of units backordered or lost is a measure of the impact of the stockout. It should be noted that since the safety stock is defined as the *average net* stock just before a replenishment arrives, its numerical value is influenced by whether backordering is actually possible.

Continuous vs Periodic Review. A key question in inventory control systems is: "how often should the inventory status be determined?". The answer

to this question specifies the review interval (R), which is the time that elapses between two consecutive moments at which we know the stock level.

An extreme case is the so called "continuous review". Under continuous review the stock status is always known. In reality, continuous surveillance is almost never employed; instead, each transaction (shipment, receipt, demand, etc.) typically triggers an immediate updating of the status (*transaction reporting*).

Under "periodic review", as the name implies, the stock status is determined only every R time units; between the moments of review there may be considerable uncertainty concerning the value of the stock level.

Example 1.2.5. A common example of periodic review system is the petrol station. The drivers of the gas truck comes regularly, say once every other day, to refill the station. If the station runs out of gas between two visits, no action is taken until the next review. ◇

Inventory control policies. The *form* of the inventory control policy is tightly related to the following two issues: "When should a replenishment order be placed?" and "How large should the replenishment order be?". There are a number of possible control systems, in what follows we shall review four possible types which are rather common in practical applications. The notation we will use is the following: s denotes a reorder point, which is the inventory position threshold which triggers a replenishment; Q denotes a fixed order quantity; S denotes the order-up-to-level, that specifies a level to which the order issued should bring the current inventory position.

Order-Point, Order Quantity (s, Q) Policy: This is a continuous review policy (that is, $R = 0$) that results extremely simple to be implemented in practice. A fixed quantity Q is ordered whenever the inventory position drops to the reorder point or lower (Fig. 1.3). It should be noted that the inventory position, and not the net stock, is used to trigger an order. This because the inventory position includes the on-order stock and it takes proper account of the material requested but not yet received.

Order-point, Order-Up-to-Level (s, S) Policy: Again this is a continuous review policy where a replenishment is made whenever the inventory position drops

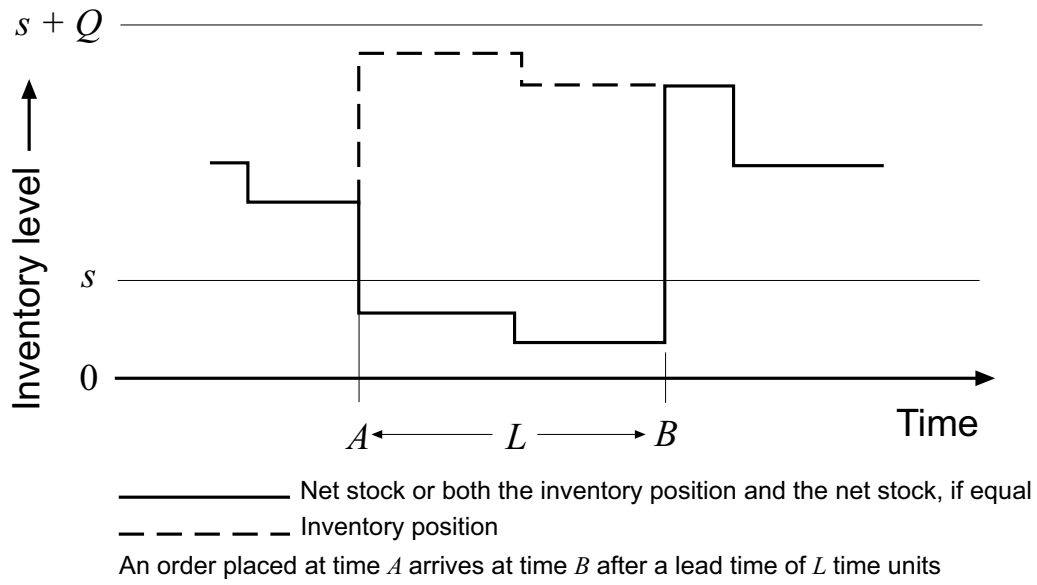


Figure 1.3: The (s, Q) system

to the order point s or lower (Fig. 1.4). In contrast to the (s, Q) policy, here a variable replenishment quantity is used, in fact we order enough to raise the inventory position to the order-up-to-level S. If all demand transactions are unit-sized, the two systems are identical because the replenishment decision will always be made when the inventory position is exactly s, that is $S = s + Q$. Otherwise if transactions larger than unit-size are allowed the replenishment quantity in the (s, S) system becomes a variable. It should be noted that the best (s, S) policy can be shown to have total cost of replenishment, carrying inventory, and shortage no larger than those of the best (s, Q) policy. However, the computational effort to find the *best* (s, S) pair is substantially more.

Periodic-Review, Order-Up-to-Level (R, S) Policy: This policy, also known as replenishment cycle policy, is in common use especially when items are ordered from the same supplier, or require resource sharing. Every R units of time (that is, at each review instant) we order the amount required to raise the inventory position to the level S (Fig. 1.5).

(R, s, S) Policy: This policy combines (s, S) and (R, S). The idea is that every R units of times we check the inventory position. If it is at or below the reorder point s, we order enough to raise it to S. If the position is above s, nothing is

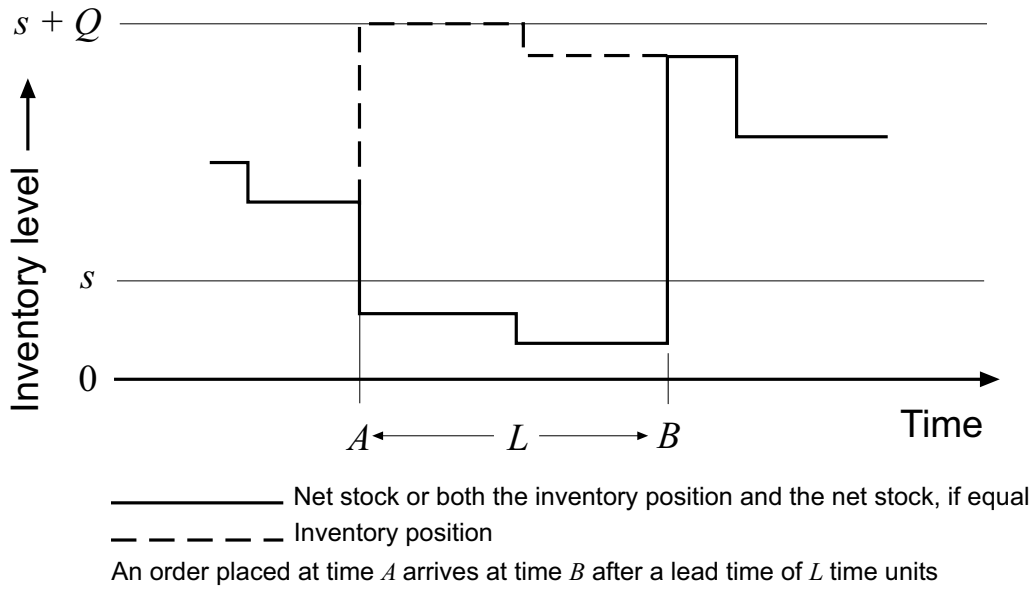


Figure 1.4: The (s, S) system

done until at least the next review. The (s, S) policy is the special case where $R = 0$, and the (R, S) is the special case where $s = S - 1$. Alternatively one can think of the (R, s, S) as a periodic implementation of (s, S) with $s = S - 1$. It has been shown [76] that under quite general assumptions on the demand pattern and the cost structure, the best (R, s, S) policy produces a lower total cost than any other policy. Nevertheless the computational effort to find the optimal policy parameters R, s and S is more intense than for any other policy.

We presented four inventory control policies of common use. It should be noted that demand uncertainty is not the only reason for which we may not be able to satisfy some of customers' demand on a routine basis directly out of stock. When the supplier capacity or the replenishment lead-time — the time required for the items ordered to be effectively available on the shelf — are probabilistic, we may also end up at some point without enough items to satisfy all the demand. As we have seen, under all these possible sources of uncertainty, if demand is unusually large, lead-times are longer than expected or we are operating for some reason at reduced capacity, a stockout may occur or emergency actions may be required to avoid a stockout. On the other hand, if demand is lower than anticipated

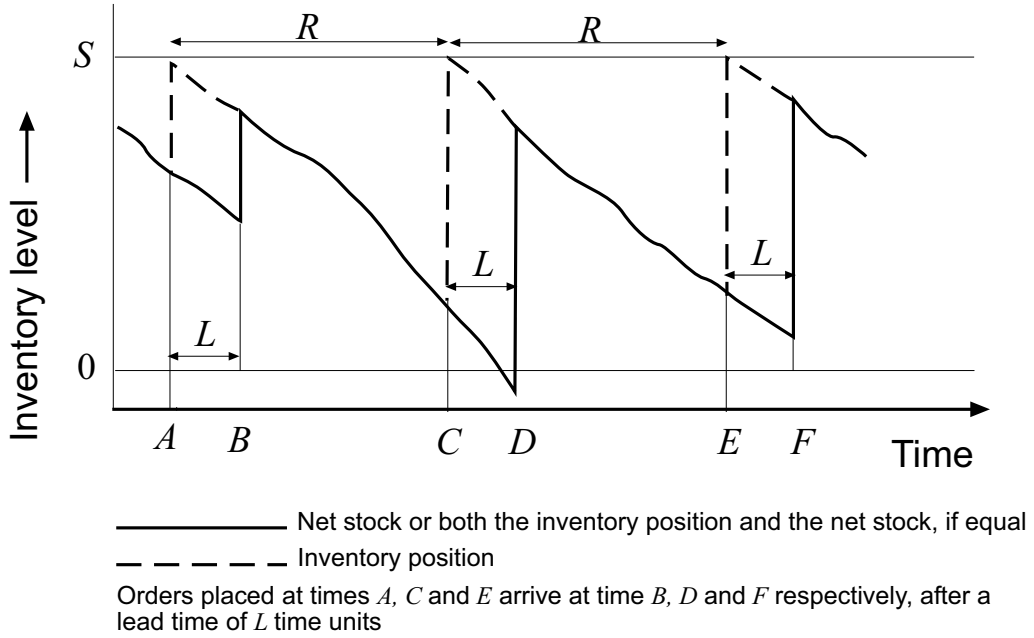


Figure 1.5: The (R, S) system

or lead-times are shorter than expected, then the replenishment arrives earlier than needed and inventory is carried at a cost. Safety stocks are the main lever to hedge against uncertainty. Different perspectives can be adopted to balance these two types of risk.

Safety Stocks Based on Minimizing Cost: These approaches involve specifying a way of costing a shortage and then minimizing the total cost of ordering, carrying inventory and dealing with shortages. Holding more inventory reduces the probability of shortages, but increases the inventory holding cost. The objective is to then find the optimal trade off that minimizes the overall cost.

Safety Stocks Based on Customer Service: Often it is the case that costing shortages raises difficulties. An alternative approach adopted by the management is then to introduce a control parameter known as *service level*. The service level becomes a constraint in establishing the safety stock of an item; for example, minimize the carrying cost of an item subject to satisfying, routinely from stock, 95% of all demands. There is a considerable choice in the selection of a service measure. Three commonly used measures are the *cycle service level*, the *fill-rate*, and the *ready-rate*. The cycle (or α) service level denotes the required minimum

fraction of cycles in which a stockout does not occur. A stockout is defined as an occasion when the on-hand-stock drops to the zero level. The fill-rate (or β service level) is the fraction of customer demand that is met routinely; that is, without backorders or lost sales. Finally, the ready-rate is the fraction of time that net stock is positive.

1.3 Related works

In this section we discuss related works in three areas: SCP, stochastic inventory control and hybrid methods employing techniques from Operations Research and CP for combinatorial optimization. Related works in SCP are discussed in Section 1.3.1. Previous approaches to stochastic inventory control are discussed in Section 1.3.2. Works on integration between CP and Operations Research are discussed in Section 1.3.3.

1.3.1 Stochastic Constraint Programming

In this section we discuss, firstly, the seminal work on SCP by Walsh [98]. Secondly we discuss two solution techniques that have been proposed and that build two alternative solution methods on the original framework proposed by Walsh: a scenario based approach by Tarim et al. [91]; and an improved forward checking procedure and an arc consistency algorithm by Balafoutis and Stergiou [5].

Foundations. To the best of our knowledge the first work that tried to create a bridge between Stochastic Programming and CP is by Benoist et al. [9]. This work is mainly a review over existing Stochastic Programming techniques for optimization under uncertainty and ad-hoc approaches developed by the CP community to cope with similar problems. The authors emphasize the fact that, while Stochastic Programming [11] produced a wealth of impressive results over the last 35 years, in the CP community people often developed and used ad-hoc techniques, of which very little has been formalized. The authors mention, among the typical approaches adopted in CP for optimization under uncertainty, the use of a static combinatorial algorithm using expected values as inputs, the use of simulation-based optimization to compare possible decisions, and finally the use of hybrid approaches trying to introduce the stochastic nature in the general design of the algorithm. The authors also conclude that CP, because of its expressing power, is particularly suitable for modeling and solving combinatorial optimization problems that are stochastic in nature. Both simulation-based approaches for optimization under uncertainty and expected value-based approaches can be easily implemented, nevertheless the authors left several questions opened: “how

can the search space be abstracted from a stochastic description, onto which a combinatorial approach can be found?"; "how to obtain upper (lower) bounds to be used in a branch-and-bound algorithm?"; "how can a fast and incremental CP simulation engine be built, which possibly integrates hybrid methods combining CP and Stochastic Programming methods?". Some of these questions have been addressed in some recent works by Walsh [98], Tarim et al. [91], Balafoutis and Stergiou [5] etc. Walsh [98] proposed *Stochastic Constraint Programming*, a generic framework for representing problems that are stochastic in nature using CP. Walsh [98] and Tarim et al. [91] proposed two effective and alternative ways for representing the search space of a generic stochastic constraint program. Some other questions will be answered in this work. For instance, how to obtain and exploit tight upper (lower) bounds through Stochastic Programming techniques to perform cost-based filtering for certain classes of stochastic constraint programs. Other questions are still open, particularly those concerning the integration of efficient general purpose techniques for stochastic optimization in CP.

The framework. SCP is a framework proposed by Walsh [98]. The framework has been described in Section 1.2.3 and it is meant to model decision problems involving uncertainty and probability. In contrast to CP, SCP features both decision and *random* (or stochastic) variables. Walsh discusses both the semantics of this framework and the computational complexity of a generic stochastic constraint program. He also proposes two complete algorithms in [98] for solving stochastic constraint program: a **backtracking algorithm** and a **forward checking** algorithm.

The backtracking algorithm differentiates between decision and stochastic variables. On meeting a decision variable, it tries each value in its domain in turn. On meeting a random variable, it tries each value in turn and it returns the sum of the answers to the subproblems weighted by the probabilities of their occurrence. The algorithm also follows a scheme similar to the Davis-Putnam like algorithm for stochastic satisfiability [61], employing upper and lower bounds on satisfaction probability for a given random variable assignment to prune search and determine optimal satisfaction.

The forward checking algorithm is based on the backtracking algorithm. On

instantiating a decision or a random variable, it checks forward and it prunes value from the domains of future decision and stochastic variables which break constraints. Walsh also briefly mentions some approximation procedures, namely a strategy where stochastic variables are replaced by their most probable values, thus leading to a deterministic constraint satisfaction problem, and a strategy employing Monte Carlo sampling to test a subset of the possible worlds.

There are three main assumptions in the framework proposed by Walsh. Firstly, his framework assumes that stochastic variables are independent, instead in the work on Tarim et al. [91] dependency between random variables will be properly accounted by means of *scenarios* and effective sampling techniques. Secondly, probability distributions are not allowed to change over time and are assumed to be fixed and known a-priori. Thirdly, variable domains are assumed to be finite, this third assumption will be in some cases relaxed in our work, thus allowing continuous distributions to be considered.

Walsh also discusses related works that inspired SCP. Both stochastic integer programming [11] and stochastic satisfiability [61] originally motivated SCP. SCP shares the advantages that CP has over integer programming and over satisfiability (eg. global constraints, non-linear constraints, and constraint propagation). Mixed constraint satisfaction [29] is closely related to one-stage stochastic constraint programs. In [79] constraint satisfaction has been extended to include probabilistic preferences on the values assigned to variables. Branching constraint satisfaction [35] models problems in which there is uncertainty in the number of variables. Walsh also points to three existing extensions of the traditional constraint satisfaction problem that model uncertain constraints. *Partial constraint satisfaction* [36] tries to maximize the number of constraints satisfied. *Probabilistic constraint satisfaction* [29] assigns to each constraint a certain probability of being part of the problem, this probability is independent of all the other constraints that participate to the problem. Finally *valued* and *semi-ring based constraint satisfaction* [12] generalizes probabilistic constraint satisfaction in the sense that a value is associated with each tuple in a constraint, whilst in valued constraint satisfaction, a value is associated with each constraint. Nevertheless Walsh points out the fact that none of these approaches deal with variables that may have uncertain or probabilistic values as SCP does.

A scenario-based approach. In [91] Tarim et al. proposed scenario based SCP. The novelty in this work is the fact that the authors adopt a semantics for stochastic constraint programs based on scenario trees. By using this semantics the authors can compile stochastic constraint programs into conventional (non-stochastic) constraint programs and they can therefore use existing constraint solvers to effectively solve this class of problems.

Scenario-based SCP has been outlined in Section 1.2.3. Tarim et al. not only defined a general way to compile stochastic constraint programs into conventional constraint programs, but they also proposed a language, that is stochastic OPL, which is based on the OPL constraint modeling language [46]. Using this language the authors modeled optimization problems under uncertainty from a variety of fields: portfolio selection; agricultural planning; and production/inventory management.

The main novelty brought by this scenario based approach is the fact that it allows multiple chance-constraints and a range of different objectives to be modeled, such as Markowitz's mean/variance model. The authors point out that each of these changes would require substantial modifications in the backtracking and forward checking algorithms proposed in [98]. The scenario based view allows each of these extension to be easily modeled using stochastic OPL, compiled down into standard OPL and solved by means of existing solvers. It should be noted that the approach is general and the compilation does not need necessarily to be performed using OPL, but it can be implemented using any available CP language and/or software package. The main drawback of this approach is related to the fact that the scenario tree required to model a given problem exponentially grows in size when random variable domains are large thus leading to large models difficult to be solved. However, the authors in [91] remark that a scenario-based approach is feasible for many problems and that they observed much better performance using scenario-based approach on the book production planning example of Walsh [98] compared to the tree search methods.

In addition to this general purpose modeling/solving framework the authors also proposed some technique to improve the efficiency of the solution process. In order to do so, they proposed scenario reduction techniques, such as Monte Carlo Sampling or Latin Hypercube Sampling [84], to reduce the number of scenarios

considered in the model. Their experimental results show the effectiveness of this approach, which in practice is able to find high quality solutions using a small number of scenarios. Finally, inspired by robust optimization techniques used in Operations Research [60], the authors also proposed some techniques to generate robust solutions, that is solutions that adopt similar (or the same) decisions under different scenarios.

An improved forward checking procedure and an Arc Consistency algorithm. The scenario based semantics of Tarim et al. for SCP is a valid alternative to the original policy based semantics proposed by Walsh. The policy based semantics in [98] has been further explored in [5]. In this work Balafoutis and Stergiou propose an improved formulation for the original forward checking procedure proposed by Walsh.

The new forward checking procedure takes better advantage of probabilities and achieves stronger pruning. The key observation is related to the fact that when a forward check is operated and values from future stochastic variables are removed, the strategy in [98] exploits only a "local" view of the future problem. Thus it is not taken into account the fact that, as values are removed from future stochastic variables, the maximum possible satisfaction of the current assignment is reduced. In other words the strategy in [98] considers value removals from any future stochastic variable as "independent" of value removals from other future stochastic variables.

In addition to the improved forward checking strategy the authors in [5] also define arc consistency for stochastic constraint programs, in analogy with the widely known notion of arc consistency [1] for classic (deterministic) constraint programs, which we discussed in Section 1.2.2. Based on this definition an arc consistency algorithm is proposed that is able to handle constraints of any arity and that is particularly effective on binary constraints. Furthermore, a Maintaining Arc Consistency algorithm is also proposed, that can operate on non-binary problems.

1.3.2 Stochastic Inventory Control

In this section we discuss the relevant literature on stochastic inventory control and in particular on the (R,S) model, to which we extensively apply SCP techniques in this dissertation.

In Section 1.2.4 the relevant formal background on deterministic/stochastic inventory control and stochastic lot sizing has been provided. For a further discussion the reader can refer to several textbooks on inventory theory [53, 64, 81, 102]. Although an extensive literature exists on inventory control, this is still a very active research area especially when modeling requires uncertainty to be taken into account. Girlich and Chikan [39] give a very interesting "historical" review on the topic. A well known review on the literature on quantitatively-oriented approaches for determining lot sizes when production or procurement yields are random is provided by Yano and Lee [100]. Yano and Lee underline the fact that very little literature exists on multi-period stochastic lot sizing problems.

An interesting class of production/inventory control problems considers the single-location, single-product case under non-stationary stochastic demand. In this class of problems a fixed procurement cost is charged each time a replenishment order is placed, whatever the size of the order, and a linear holding cost is charged on any unit carried over in inventory from one period to the next. The objective is to minimize the expected total cost under a service level constraint, that is the probability that at the end of every time period the net inventory will not be negative or a penalty cost incurred for each unit of demand that is back-ordered. This class has been widely studied because of its key role in practice.

As discussed in section 1.2.4 one of the possible policies that can be adopted to cope with this class of problems is the *replenishment cycle policy* or (R,S) policy. A detailed discussion on the characteristics of (R,S) can be found in [22]. We recall that in this policy a replenishment is placed every R periods to raise the inventory level to the order-up-to-level S. This provides an effective means of damping planning instability (deviations in planned orders, also known as *nervousness* [23, 44]) and coping with demand uncertainty. As pointed out by Silver et al. ([81], pp. 236–237), (R,S) is particularly appealing when items are ordered from the same supplier or require resource sharing. In these cases all items in a co-

ordinated group can be given the same replenishment period. In [51] Janssen and de Kok discuss a two-supplier periodic model where one supplier delivers a fixed quantity while the amount delivered by the other is governed by an (R,S) policy. In [82] Smits et al. consider a production-inventory problem with compound renewal item demand. The model consists of stock-points, one for each item, controlled according to (R,S)-policies and one machine which replenishes them. Periodic review also allows a reasonable prediction of the level of the workload on the staff involved, and is particularly suitable for advanced planning environments and risk management [85]. For these reasons (R,S) is a popular inventory policy.

Under the assumption of non-stationary demand the (R,S) policy takes the form (R^n,S^n) where R^n denotes the length of the n^{th} replenishment cycle and S^n the corresponding order-up-to-level.

For the service level constrained problem, early works were heuristic (Silver [80] and Askin [3]). Bookbinder and Tan [15] proposed another heuristic, under the static-dynamic uncertainty strategy. In this strategy, the replenishment periods are fixed at the beginning of the planning horizon and the actual orders at future replenishment periods are determined only at those replenishment periods, depending upon the realized demand.

For the formulation under penalty cost scheme a mixed integer non-linear program has been proposed by Sox [83]. A solution algorithm that resembles the Wagner-Whitin [96] algorithm but with some additional feasibility constraints has been also presented in the same work.

The first complete approach for solving the non-stationary (R,S) policy under service level constraints has been proposed by Tarim and Kingsman in [89]. This approach operates under mild assumptions and models the problem as a mixed integer linear program. The model proposed can be solved by means of any available off-the-shelf tool such as ILOG CPLEX [49].

Similarly, a mixed integer program — which again operates under similar mild assumptions — has been proposed by Tarim and Kingsman in [90] for the formulation operating under a penalty cost scheme. In this case the cost function in the Stochastic Programming formulation of the problem is non-linear and it cannot be directly represented in the mixed integer linear program. This function is therefore modeled by means of a piecewise linear approximation. Again, the model

provided can be solved using any available package for mathematical programming.

Both the two models discussed in the former paragraphs are very effective and provide for the first time two practical means for computing near-optimal policy parameters for the replenishment cycle policy.

In [92] an efficient CP formulation has been proposed by Tarim and Smith for the service level constrained problem. This formulation exploits key features of CP: search heuristics, global constraints and discrete domains. The search process is guided in such a way to branch first on binary variables. The model is formulated in a more natural way than the respective mixed integer program, by employing decision variables to index other decision variables. A preprocessing algorithm is proposed to reduce a-priori the set of optimal candidate values in decision variable domains and thus to reduce the effort spent in the tree-search process. The model proves to be much more effective than the respective mixed integer programming formulation.

Two recent works by Tempelmeier [93] and by Pujawan and Silver [65] show that finding optimal replenishment cycle policy parameters — in both a heuristic or a complete way — is an active research area and prove the interest that the works by Tarim and Kingsman raised in the Operations Research community. Specifically the first work extends Tarim and Kingsman model under service level constraints in order to cosider a different service level measure, the β service level (or fill rate), which has been discussed in Section 1.2.4. The second work develops two heuristics to minimize the expected total relevant cost per unit time. These heuristics try to select an appropriate augmentation quantity beyond the expected total demand through to the planned (deterministic) time of the next replenishment.

1.3.3 Integration of Operations Research and Constraint Programming Techniques in Combinatorial Optimization

In this section we shall give a brief overview on the integration of Operations Research and CP techniques in combinatorial optimization. This research area is attracting more and more attention in different communities. An extensive dis-

cussion on hybrid techniques for combinatorial optimization is presented in [94]. The discussion in [94] is mainly focused on integrating CP and mathematical programming (and in particular mixed integer linear programming) for combinatorial optimization. This is only one of the many possible directions for integrating CP with other techniques from Operations Research and Artificial Intelligence. For instance, CP has been successfully integrated with local search [99], DP [33], linear programming and cost-based reasoning [31, 32]. All these works show that techniques from Operations Research and Artificial Intelligence can be effectively incorporated within global constraints in constraint programs in order to achieve stronger filtering during the search, guide the search process, quickly obtain near-optimal solutions, etc.

For the discussion in this dissertation, it is particularly interesting to further describe the approaches in [31, 32] and [33].

In the first work, by Focacci et al. [31, 32], a linear programming relaxation is employed in the filtering process. The filtering is performed using the reduced costs provided in the final tableau that gives the solution of the linear program. Nevertheless the approach described in their work is general and does not necessarily need reduced costs or a linear relaxation to be performed. In fact, as already discussed, *optimization-oriented global constraints* embed a generic optimization component, representing a proper relaxation of the constraint itself, into a global constraint.

In the second work, by Focacci and Milano [33], the original combinatorial optimization problem of interest, typically NP-hard, is relaxed in such a way to obtain a new problem whose DP state space representation [20] contains a number of nodes and arcs polynomial in the problem input. The solution to this relaxed problem is efficiently obtained using a shortest path algorithm in the state space. The optimal solution to this relaxed problem provides a bound that is again used for filtering purposes or for guiding the search.

1.4 Thesis Statement

In this section, firstly we provide a summary of the work described in this dissertation, secondly we highlight the contributions of our work. Finally, for each of the following chapters we summarize the respective content.

1.4.1 Summary

This dissertation is mainly focused on investigating the application of SCP techniques in the area of stochastic inventory control. Hybrid techniques integrating SCP with DP and other approaches borrowed from Operations Research are employed for improving the optimization process.

We concentrate on an interesting problem of practical interest in inventory control: the computation of optimal replenishment cycle policy parameters under non-stationary stochastic demand. As discussed in Section 1.3.2, this problem has been the object of significant research in the last twenty years. We consider different existing formulations of this problem, namely the one under service level constraints, and the one under penalty cost scheme. For both these formulations the existing approaches proposed by Tarim and Kingsman [89, 90] present two drawbacks.

Firstly, these approaches are not complete and can provide only near-optimal solutions. Specifically, for both the models mixed integer linear programs have been proposed. The one proposed to address the service level constrained problem [89] assumes that negative orders are not allowed, so that if the actual stock exceeds the order-up-to-level for that review, this excess stock is carried forward and not returned to the supply source. This event is assumed to be rare, and therefore its effects are ignored. As a direct consequence of this, the model only computes suboptimal policy parameters and an approximate expected total cost. The model proposed under penalty cost scheme [90] operates under the same assumption, but in addition to that it also employs a piecewise linear approximation for representing the cost function.

Secondly, these approaches do not scale well and perform poorly for real-world sized instances. Specifically, both the models require a large number of

binary decision variables and, in addition to this, the model under penalty cost scheme quickly becomes intractable as the planning horizon length and the number of segments in the piecewise linear approximation increase.

Furthermore, no approach in the literature exists for computing optimal replenishment policy parameters under non-stationary stochastic demand when a stochastic delivery lag is considered for each order issued. Indeed a model that considers immediate delivery is a poor representation of the real world.

In our work, firstly we address the assumption on negative orders for the service level constrained model. In order to do so, we develop a novel modeling tool in SCP — *global chance-constraints* — that lets us fully represent the complex interactions that arise when multiple chance-constraints are added in a model. One of the conclusions drawn is that the original assumption tends to underestimate holding costs and to produce, in certain cases, buffer stocks higher than strictly necessary. Nevertheless in general this assumption does not significantly affect the quality of the optimal policy parameters computed. Therefore, when considering the problem under penalty cost scheme, we retain the assumption on negative orders, and we employ global chance-constraints to represent the non-linear cost function and to obtain a more accurate solution than the one provided by the mixed integer linear program.

Global chance-constraints have been employed in our work not only to obtain more accurate or complete solutions, but also to obtain more efficient reformulations of the existing models. Specifically, we enhanced the SCP model proposed by Tarim and Smith [92] by augmenting it with three global chance-constraints implementing dedicated cost-based filtering techniques. We also enhanced with similar techniques our SCP model under penalty cost scheme.

Finally we employed global chance-constraints to represent multiple level of uncertainty, namely demand uncertainty and delivery uncertainty, and compute optimal policy parameters for this challenging model that so far has not been studied in the literature.

In the next section we analyze in details the contributions of this work.

1.4.2 Contributions

From a theoretical point of view there are two main contributions in this dissertation: we introduced the novel concepts of *global chance-constraints* and of *optimization-oriented global chance-constraints*. The first, as stated, let us model complex interactions that arise in stochastic constraint programs where several chance-constraints appear together. The second let us apply cost based filtering in a stochastic environment, by exploiting cost-based reasoning and/or relaxations involving decision variables, random variables and the constraints defined on these.

From a practical point of view, our contribution consists in the application of both these techniques to known problems in the area of stochastic inventory control.

Global chance-constraints

There are three main contributions related to this novelty:

- **Formal background.** We have formally introduced *global chance-constraints*, defined as constraints that capture a relation among a non-fixed number of decision and random variables. These constraints not only are more expressive than the respective aggregation of simple chance-constraints, but they can be associated with more powerful filtering algorithms (Chap. 2).

- **Application 1.** We have applied *global chance-constraints* to compute optimal replenishment cycle policy parameters under non-stationary stochastic demand and service level constraints. *Global chance-constraints* allow the assumption on negative orders adopted in previous works [89, 92] to be relaxed and thus they let us compute the real optimal solution for the problem (Chap. 2).

- **Application 2.** We exploited *global chance-constraints* to represent multiple layers of uncertainty, demand uncertainty and delivery uncertainty, and to compute replenishment cycle policy parameters under non-stationary

stochastic demand, service level constraints and stochastic delivery lag (Chap. 3).

Optimization-oriented global chance-constraints

There are two main contributions related to this novelty:

- **Formal background.** We have formally introduced *optimization-oriented global chance-constraints*, defined as global chance-constraints that encapsulate suitable relaxations of the constraints considered. This relaxation, in contrast to conventional optimization-oriented global constraints, may involve stochastic variables (Chap. 4).

- **Application 3.** By using *optimization-oriented global chance-constraints*, we have augmented the SCP model originally proposed by Tarim and Smith [92] for computing optimal replenishment cycle policy parameters under non-stationary stochastic demand and service level constraints. In Tarim and Smith's model domain filtering was originally performed only in a proactive way before starting the search process. The cost-based filtering dynamically performed during the search by the optimization-oriented global chance-constraints proposed let us now efficiently compute near-optimal replenishment cycle policy parameters under non-stationary stochastic demand and service level constraints (Chap. 5). The augmented model produces run times that are orders-of-magnitude lower than those achieved by the state of the art approach in [92].

A global perspective

Finally we have employed both *global chance-constraints* and *optimization-oriented global chance-constraints* to obtain the state of the art approach for computing replenishment cycle policy parameters under non-stationary stochastic demand and a penalty cost scheme:

- **Application 4.** We have applied *global chance-constraints* to model the non-linear cost function that is only approximated by the approach in

[90], which employs a piecewise linear approximation for modeling period holding and back-ordering costs. In addition to this we have applied *optimization-oriented global chance-constraints* to the same model in order to perform cost-based reasoning and thus improve the efficiency of the search process (Chap. 6).

1.4.3 Paper I (Chap. 2): A Global Chance-Constraint for Stochastic Inventory Systems under Service Level Constraints [75]

SCP has been introduced in [98] to model decision problem involving uncertainty and probability. In contrast to conventional approaches in Stochastic Programming, SCP features all the key features of CP: constraint propagation, variable and value selection strategies and so forth.

To solve stochastic constraint programs, Tarim et al. in [91] proposed a semantics based on scenario trees. This semantics is extremely flexible, especially for the fact that it lets stochastic constraint programs be compiled down into conventional constraint programs, so that conventional constraint solvers can be employed to find a solution. Nevertheless, the framework proposed by Tarim et al. still presents limits: in particular, as formulated in [91], it does not specify how a generic relation among a non-predefined number of decision variables and stochastic variables under a given policy of response should be translated into a conventional constraint program. This is obviously not an easy task, as it is problem dependent.

In order to address this issue we propose in this chapter an extension for SCP: *global chance-constraints*. Global chance-constraints, similarly to conventional global constraints, represent relations among a non predefined number of variables and incorporate dedicated filtering algorithms. In contrast to conventional global constraints, global chance-constraints represent relations among decision and *stochastic* variables and can model any policy of response.

By means of this novelty and using the scenario based semantics proposed by Tarim et al. [91], in this work we were able to relax the original assumption on negative order quantities that had to be adopted in [89, 92] for computing re-

plenishment cycle policy parameters under non-stationary stochastic demand. In contrast to models previously proposed our model provides (i) the exact cost of an optimal solution, and (ii) exact policy parameters, that is replenishment cycle lengths and order-up-to-levels. A comparison among our approach and previous approaches shows that the discussed assumption does not significantly affect the quality of the policy parameters computed by the models in [89, 92], but it does affect the computed cost, which typically differs significantly from the real cost of the solution provided.

1.4.4 Paper II (Chap. 3): Computing Replenishment Cycle Policy under Non-stationary Stochastic Lead Time [72]

Also in this chapter we rely on the scenario based semantics originally proposed in [91]. The problem here is to compute replenishment cycle policy parameters under non-stationary stochastic demand, delivery lag and service level constraints. Incorporating a delivery lag in inventory control models is a very active research topic, as the literature review presented in this chapter will show. To the best of our knowledge, this is the first work in which a non-stationary stochastic demand and a non-stationary stochastic delivery lag are considered together when computing replenishment cycle policy parameters under service level constraints.

The first part of this work is dedicated to the derivation of a mathematical model for computing feasible buffer stocks under non-stationary stochastic demand, delivery lag and service level constraints. The expression obtained represents a non-linear relation among decision variables (replenishment decisions and inventory levels) and stochastic variables (stochastic demands and delivery lags).

Using the expression derived in the first part of this chapter, we developed a *global chance-constraint* and the respective filtering procedure able to take into account both demand and delivery lag uncertainty while computing buffer stocks required to guarantee the given minimum service level in terms of non-stockout probability. The approach was tested against different delivery lag distributions. The experimental results presented show the behavior of the expected total cost of the optimal policy with respect to the expected value and to the variance of the delivery lag.

1.4.5 Paper III (Chap. 4): Cost-based filtering for stochastic constraint programming [74]

In this chapter we introduce *optimization-oriented global chance-constraints*. These are global chance-constraints incorporating an optimization component that allows cost-based reasoning to be performed during the search. Cost-based reasoning lets the solver filter in a proactive way provably suboptimal values from the domain of decision variables. In contrast to conventional optimization-oriented global constraint, in optimization-oriented global chance-constraints the cost-based reasoning may involve stochastic variables in different ways: by relaxing some of the constraints in which they appear, or by exploiting known inequalities borrowed from Stochastic Programming.

In this chapter we discuss a general purpose procedure for performing cost-based reasoning for certain classes of stochastic constraint programs, when some assumptions are respected. These assumptions are generally respected in practical applications, as witnessed by a large literature available in the Stochastic Programming community that operates under the same assumptions.

Two problems from the Stochastic Programming literature are considered in order to show the effectiveness of cost-based reasoning in SCP. The static stochastic knapsack problem [56] and the stochastic sequencing problem under release time and deadline, a stochastic generalization of a known NP-hard problem [37]. Our experimental results show order-of-magnitude improvements for both the problem considered.

1.4.6 Paper IV (Chap. 5): Cost-based Filtering Techniques for Stochastic Inventory Control under Service Level Constraints [87, 88]

The assumptions discussed in Chapter 4, required in order to apply the cost-based filtering strategy there discussed, are not always respected by stochastic constraint programs. When the relaxations and the inequalities there discussed cannot be applied, it is usually still possible to perform cost-based reasoning by employing some ad-hoc methodology for the problem modeled. It may also often be the

case, that even if the methods discussed are applicable, ad-hoc methodologies may provide tighter bounds and therefore be more appropriate to perform cost-based reasoning.

We consider the problem of computing replenishment cycle policy parameters under non-stationary stochastic demand and service level constraints as formulated in [89, 92]. The model proposed by Tarim and Smith is a one-stage stochastic constraint program addressed through ad-hoc techniques adopted to compute minimum buffer stocks required to meet the given service level constraints. Some ad-hoc domain filtering techniques are proposed in [92]. These techniques consider the probability distribution of the stochastic variables and the input parameters of the problem (holding cost, ordering cost, service level probability) in order to perform a preprocessing of decision variable domains based on cost-based reasoning.

In this work, in order to enhance the search process, we developed dedicated *optimization-oriented global chance-constraints* (or for simplicity, global constraints) able to dynamically perform Tarim and Smith's cost-based reasoning involving decision and stochastic variables during the search process. On the top of this we developed novel ad-hoc cost-based reasoning techniques for Tarim and Smith's model. These techniques are incomparable with those proposed by Tarim and Smith in terms of filtering power. Finally an effective DP relaxation is proposed, which can produce tight bounds employed to prune suboptimal nodes of the search tree during the search. According to what discussed in Chapter 4 also in this case experimental results show order-of-magnitude improvements with respect to both the mathematical programming formulation in [89] and the CP formulation in [92].

1.4.7 Paper V (Chap. 6): Constraint Programming for Stochastic Inventory Systems under Shortage Cost [71, 73]

This final chapter is particularly interesting since both the techniques described in former chapters, *global chance-constraints* and *optimization-oriented global chance-constraints*, are employed in order to provide the state of the art approach, both in terms of quality of the solution provided and efficiency of the search pro-

cess, for computing replenishment cycle policy parameters under non-stationary stochastic demand and a penalty cost scheme.

Global chance-constraints are employed in this chapter to dynamically compute during the search process the non-linear cost function of the problem, which in [90] was approximated by using a piecewise linear representation.

Optimization-oriented global chance-constraints are employed to perform cost-based reasoning exploiting a DP relaxation similar to the one discussed in Chapter 5, for the service level constrained problem.

For this reason this chapter somehow provides a global view on the contributions of this dissertation, since it synthesizes both the novelties proposed in a single application.

Our experimental results show: (i) the improvement in terms of quality of the solution obtained over the mixed integer linear programming model in [90], (ii) the efficiency of our approach that can be effectively applied to planning horizons of a significant length, (iii) the stability of the performances achieved under different input parameters and random demand patterns.

1.5 Future Work

Several topics in this dissertation suggest directions for future research. In this section, for each of the following chapters, we try to summarize which questions remain open and which future research directions are promising.

Chapter 2. We recall that this chapter deals mainly with the concept of *global chance-constraint*. There are both theoretical and practical aspects that should be considered in the future research on global chance-constraints. Obviously there is a clear opportunity for proposing a full family of global chance-constraints with dedicated consistency and filtering rules similarly to what has been done in the last 20 years for deterministic constraints. In our specific application discussed in Chapter 2 DP is used in the filtering procedure. We employed a trivial recursive implementation, that is obviously quite inefficient, more efficient procedures may be developed by trading space with time and by storing information in *dynamic* tables updated through a publish-subscribe mechanism triggered by constraint propagation. We believe this is a promising research direction that should be pursued in future works, since it provides a general purpose approach to deal with propagation in global chance-constraints.

Chapter 3. In this chapter a global chance-constraint is developed with the respective filtering procedure. Again we see an opportunity here for employing *dynamic* tables in order to improve efficiency as discussed above. In addition to this, we also think that the hybrid technique here employed, which merges deterministic equivalent modeling [18] and scenario based approach [11, 91], may be employed as a general technique to develop propagation algorithms for other global chance-constraints. Furthermore no bounding or filtering techniques have been discussed. It is clear that, by incorporating dedicated filtering algorithms, the proposed model has the potential of becoming very efficient. Tight bounds may be obtained, for instance, by applying the technique discussed in Chapter 4.

Chapter 4. This chapter proposes a general approach for performing cost-based filtering in SCP. Obviously the approach may have a wide range of applications that should be considered in future works. Possible research directions may also: consider different inequalities that may be suitable for generating valid bounds in the filtering process; discuss the cost-based filtering strategy when generic chance-

constraints and stochastic constraint programs are considered — the discussion in this chapter is restricted to special classes of stochastic constraint programs —; and exploit the information provided by optimization-oriented global chance-constraints to define search strategies.

Chapter 5. In this chapter we develop three optimization oriented global-chance constraints for improving the search process in a stochastic constraint program that computes replenishment cycle policy parameters under non-stationary stochastic demand and service level constraints. The resulting model, in which these three global constraints are posted, is very efficient and provides the state-of-the-art approach for computing replenishment cycle policy parameters. Obviously CP is not the only approach that can be used to solve this problem. We also explored other research directions, in particular in the field of DP. Our preliminary experience, not discussed in this dissertation, shows that DP also provides remarkable performances and it should be further explored as a valid technique for computing replenishment cycle policy parameters. Finally, techniques similar to the those developed in this chapter may be also applied to the problems discussed in Chapter 2 and 3 for speeding up the search process.

Chapter 6. This chapter, as already discussed, summarizes all the contributions of this dissertation in a single application. Again performances are very satisfactory and they suit real world problems with long planning horizons spanning up to 38 periods. Note that with 36 periods we can plan for a year ahead with a weekly granularity. Again we see a window of opportunity in this problem for applying other techniques such as DP, but we do not have any result so far in this direction. Another possible research direction consists in considering also for this problem a stochastic lead time and in developing a propagation algorithm similar to the one developed in Chapter 3. Supplier capacity constraints may also be considered, note that this would make the problem extremely hard to be treated, thus this last extension is a particularly challenging one.

1.6 Conclusions

There are two main research areas for which this dissertation represents a contribution: *Stochastic Constraint Programming* and *stochastic inventory control*. The contributions brought to the field of SCP are mainly theoretical and they consist in the introduction of two novel modeling concepts: *global chance-constraints* and *optimization-oriented global chance-constraints*. Global chance-constraints are mainly concerned with expressiveness, although they may be also used to perform efficient propagation in SCP. In contrast, optimization-oriented global chance-constraints play a key role in achieving efficiency in the search process for stochastic constraint optimization problems. The contributions brought to the field of stochastic inventory control directly follow from the application of the former novelties to well-known problems from the inventory control literature. The computation of replenishment cycle policy parameters under non-stationary demand is a very active research topic as we have shown. We improved the state of the art approaches both in terms of quality of the solution found and in terms of computational efficiency. We also augmented the complexity of the models studied in the literature by adding multiple-layers of uncertainty (i.e. demand and delivery uncertainty), a topic that has not been explored before for the non-stationary case. In summary, not only we proposed *novel optimization models and algorithms that constitute a step forward in stochastic inventory control*, but we also made significant *theoretical contributions to a new trend of research that applies constraint reasoning* — a technique that in the last 25 years generated a remarkable amount of lore — *to optimization problems under uncertainty*.

Chapter 2

Paper I: A Global Chance-Constraint for Stochastic Inventory Systems under Service Level Constraints

R. Rossi, S. A. Tarim, B. Hnich and S. Prestwich

Abstract

We consider a class of production/inventory control problems that has a single product and a single stocking location, for which a stochastic demand with a known non-stationary probability distribution is given. Under the widely-known replenishment cycle policy the problem of computing policy parameters under service level constraints has been modeled using various techniques. Tarim & Kingsman introduced a modeling strategy that constitutes the state-of-the-art approach for solving this problem. In this paper we identify two sources of approximation in Tarim & Kingsman's model and we propose an exact *stochastic constraint programming* approach. We build our approach on a novel concept, *global chance-constraints*, which we introduce in this paper. Solutions provided by our exact approach are employed to analyze the accuracy of the model developed by Tarim & Kingsman.

2.1 Introduction

The study of lot-sizing began with Wagner and Whitin [96], and there is now a sizeable literature in this area extending the basic model to consider capacity constraints, multiple items, multiple stages, etc. However, most previous work on lot-sizing has been directed towards the deterministic case. For a general overview over deterministic lot-sizing problems the reader may refer to [30].

The practical problem is that in general many, if not all, of the future demands have to be forecasted. Point forecasts are typically treated as deterministic demands. However, the existence of forecast errors radically affects the behavior of the lot-sizing procedures based on assuming the deterministic demand situation. Forecasting errors lead both to stock-outs occurring with unsatisfied demands and to larger inventories being carried than planned. The introduction of safety stocks in turn generates even larger inventories and also more orders. It is reported by Davis [21] that a study at Hewlett-Packard revealed the fact that 60% of the inventory investment in their manufacturing and distribution system is due to demand uncertainty.

As pointed out in [40] one major theme in the continuing development of inventory theory is to incorporate more realistic assumptions about product demand into inventory models. In most industrial contexts, demand is uncertain and hard to forecast. Many demand histories behave like random walks that evolve over time with frequent changes in their directions and rates of growth or decline. Furthermore, as product life cycles get shorter, the randomness and unpredictability of these demand processes have become even greater. In practice, for such demand processes, inventory managers often rely on forecasts based on a time series of prior demand, such as a weighted moving average. Typically these forecasts are predicated on a belief that the most recent demand observations are the best predictors for future demand.

An interesting class of production/inventory control problems therefore considers the single-location, single-product case under non-stationary stochastic demand. This class has been widely studied because of its key role in practice. We assume a fixed procurement cost each time a replenishment order is placed, whatever the size of the order, and a linear holding cost on any unit carried over in

inventory from one period to the next. Our objective is to minimize the expected total cost under a service level constraint, that is the probability that at the end of every time period the net inventory will not be negative. Early works in the area were heuristic (Silver [80] and Askin [3]). Bookbinder and Tan [15] proposed another heuristic, under the static-dynamic uncertainty strategy. In this strategy, the replenishment periods are fixed at the beginning of the planning horizon and the actual orders at future replenishment periods are determined only at those replenishment periods, depending upon the realized demand. The expected total cost is minimized under the minimal service-level constraint.

We focus on the work of Tarim & Kingsman [89], where the authors proposed a mathematical programming approach to compute near-optimal policy parameters for the inventory control policy known as the *replenishment cycle policy* or (R,S) policy. A detailed discussion on the characteristics of (R,S) can be found in [22]. In this policy a replenishment is placed every R periods to raise the inventory level to the order-up-to-level S. This provides an effective means of damping planning instability (deviations in planned orders, also known as *nervousness* [23, 44]) and coping with demand uncertainty. As pointed out by Silver et al. ([81], pp. 236–237), (R,S) is particularly appealing when items are ordered from the same supplier or require resource sharing. In these cases all items in a coordinated group can be given the same replenishment period. In [51] Janssen and de Kok discuss a two-supplier periodic model where one supplier delivers a fixed quantity while the amount delivered by the other is governed by an (R,S) policy. In [82] Smits et al. consider a production-inventory problem with compound renewal item demand. The model consists of stock-points, one for each item, controlled according to (R,S)-policies and one machine which replenishes them. Periodic review also allows a reasonable prediction of the level of the workload on the staff involved, and is particularly suitable for advanced planning environments and risk management [85]. For these reasons (R,S) is a popular inventory policy. Under the assumption of non-stationary demand it takes the form (R^n,S^n) where R^n denotes the length of the n^{th} replenishment cycle and S^n the corresponding order-up-to-level.

Tarim & Kingsman's formulation operates under the assumption that negative orders are not allowed, so that if the actual stock exceeds the order-up-to-level

for that review, this excess stock is carried forward and not returned to the supply source. This event is assumed to be rare, and therefore its effects are ignored. As a direct consequence of this, the model only computes suboptimal policy parameters and an approximate expected total cost.

In this paper we exploit *stochastic constraint programming*, a novel modeling framework introduced by Walsh [98], to fully model the original stochastic programming formulation for computing (R^n, S^n) policy parameters. In our approach we extend the original framework with a new concept, *global chance-constraints*, and we employ this to compute optimal (R^n, S^n) policy parameters and the exact expected total cost for a given parameter configuration. By using optimal solutions provided by our model we gauge the accuracy of the solutions provided by Tarim & Kingsman's approach for a set of instances. In our experiments we show that the assumption adopted in Tarim & Kingsman's model are justified and that their model constitutes a valid trade-off for computing near-optimal (R^n, S^n) policy parameters when a short computational time is required.

This paper is organized as follows. In Section 2.2 we provide some formal background about different modeling techniques employed in this paper: stochastic programming, constraint programming, stochastic constraint programming and inventory control models. In Section 2.3 we review the existing approaches developed in the literature to compute (R^n, S^n) policy parameters. In Section 2.4 we introduce *global chance-constraints* and we present a novel *stochastic constraint programming* approach, based on this new concept, to compute optimal (R^n, S^n) policy parameters. In Section 2.5 we compare results produced by our exact approach with those provided by the state-of-the-art MIP approach for computing near-optimal (R^n, S^n) policy parameters. In Section 2.6 we draw conclusions.

2.2 Formal background

In this paper we employ and merge several different modeling techniques. In this section some formal background and references are given for each technique exploited.

2.2.1 Stochastic Programming

Stochastic programming [11] is a well known modeling technique that deals with problems where uncertainty comes into play. Problems of optimization under uncertainty are characterized by the necessity of making decisions without knowing what their full effect will be. Such problems appear in many application areas and present many interesting conceptual and computational challenges. Stochastic programming needs to represent uncertain elements of the problem. Typically random variables are employed to model this uncertainty to which probability theory can be applied. For this purpose such uncertain elements must have a known probability distribution. The typical requirement in stochastic programs is to maintain certain constraints, called *chance constraints* [18], satisfied at a prescribed level of probability. The objective is typically related to the minimization/maximization of some expectation on the problem costs. There are several different approaches to tackle stochastic programs. A first method dealing with stochastic parameters in stochastic programming is the so-called *expected value model* [11], which optimizes the expected objective function subject to some expected constraints. Another method, *chance-constrained programming*, was pioneered by Charnes and Cooper [18] as a means of handling uncertainty by specifying a confidence level at which it is desired that the stochastic constraint holds. Chance-constrained programming models can be converted into deterministic equivalents for some special cases, and then solved by some solution methods of deterministic mathematical programming. A typical example for this technique is given by the Newsvendor problem [81]. However it is almost impossible to do this for complex chance-constrained programming models. A third approach employs scenarios, which are particular representations of how the future might unfold. Each scenario is assigned a probability value, that is its likelihood. Some kind of probabilistic model or simulation is used to generate a batch of such scenarios. The challenge then, is how to make good use of these scenarios in coming up with an effective decision.

2.2.2 Constraint Programming

A *Constraint Satisfaction Problem* (CSP) [1, 17, 62] is a triple $\langle V, C, D \rangle$, where V is a set of decision variables, D is a function mapping each element of V to a

domain of potential values, and C is a set of constraints stating allowed combinations of values for subsets of variables in V. A *solution* to a CSP is simply a set of values of the variables such that the values are in the domains of the variables and all of the constraints are satisfied. We may also be interested in finding a feasible solution that minimizes (maximizes) the value of a given objective function over a subset of the variables. Alternatively, we can define a constraint as a mathematical function: $f : D_1 \times D_2 \times \ldots \times D_n \rightarrow \{0, 1\}$ such that $f(x_1, x_2, \ldots, x_n) = 1$ if and only if $C(x_1, x_2, \ldots, x_n)$ is satisfied. Using this functional notation, we can then define a constraint satisfaction problem (CSP) as follows (see also [1]): given n domains D_1, D_2, ..., D_n and m constraints f_1, f_2, ..., f_m find x_1, x_2, ..., x_n such that

$$f_k(x_1, x_2, \ldots, x_n) = 1, \quad 1 \leq k \leq m; \tag{2.1}$$

$$x_j \in D_j, \qquad 1 \leq j \leq n. \tag{2.2}$$

The problem is only a feasibility problem, and no objective function is defined. Nevertheless, CSPs are also an important class of combinatorial optimization problems. Here the functions f_k do not necessarily have closed mathematical forms (for example, functional representations) and can be defined simply by providing the subset S of the set $D_1 \times D_2 \times \ldots \times D_n$, such that if $(x_1, x_2, \ldots, x_n) \in S$, then the constraint is satisfied.

We now recall some key concepts in *Constraint Programming* (CP): constraint filtering algorithm, constraint propagation and arc-consistency [67]. In CP a filtering algorithm is typically associated with every constraint. This algorithm removes values from the domains of the variables participating in the constraint that cannot belong to any solution of the CSP. These filtering algorithms are repeatedly called until no new deduction can be made. This process is called propagation mechanism. In conjunction with this process CP uses a search procedure (like a backtracking algorithm) where filtering algorithms are systematically applied when the domain of a variable is modified. One of the most interesting properties of a filtering algorithm is arc-consistency. We say that a filtering algorithm associated with a constraint establishes arc-consistency if it removes all the values from the domains of the variables involved in the constraint that are not

consistent with the constraint. As a consequence of results in [70], where authors proved that any non-binary constraint can be translated into an equivalent binary one with additional variables, several studies on arc-consistency were limited to binary constraints. However modeling problems by means of binary constraints presents several drawbacks. Firstly these constraints are poor in term of expressiveness. Secondly the domain reduction achieved by the respective filtering algorithm associated is typically weak. In order to overcome both these problems constraints that capture a relation among a non-fixed number of variables were introduced. These constraints not only are more expressive than the respective aggregation of simple constraints, but they can be associated with more powerful filtering algorithms that take into account the simultaneous presence of simple constraints to further reduce the domains of the variables. These constraints are called *global constraints*. One of the most well known examples is the `alldiff` constraint [66], both because of its expressiveness and its efficiency in establishing arc-consistency.

2.2.3 Stochastic Constraint Programming

In [98] and [91] a *stochastic constraint satisfaction problem* (stochastic CSP) is defined as a 6-tuple $\langle V, S, D, P, C, \theta \rangle$, where V is a set of decision variables and S is a set of stochastic variables, D is a function mapping each element of V and each element of S to a domain of potential values. A decision variable in V is *assigned* a value from its domain. P is a function mapping each element of S to a probability distribution for its associated domain. C is a set of constraints. A constraint $h \in C$ that constrains at least one variable in S is a *chance-constraint*. θ_h is a threshold value in the interval $[0, 1]$, indicating the minimum satisfaction probability for chance-constraint h. Note that a chance-constraint with a threshold of 1 is equivalent to a hard constraint.

A stochastic CSP consists of a number of *decision stages*. Solving a stochastic CSP implies a two step process.

In the first step a *policy of response* has to be defined. A policy of response states the rules that decide when decision variables have to be set. There are two extreme policies: here-and-now and wait-and-see. The *here-and-now* policy sets

all decision variables before observing the realization of the random variables. A solution can be therefore expressed as an assignment for decision variables in V. The *wait-and-see* policy delays as much as possible the assignment of a value to a decision variable. Therefore a decision variable $x_i \in V$ is set to a value only after the realizations of stochastic variables $y_1, \ldots, y_{i-1} \in S$ have been observed. Under this policy typically the solution of a stochastic CSP is represented by means of a *policy tree* [91]. A policy tree is a tree of decisions where each path represents a different possible scenario (set of values for the stochastic variables) and the values assigned to decision variables in this scenario. Hybrid policies can be defined by stating at which stage k, $1 \leq k \leq j$ a decision variable x_j has to be set. The solution for any policy that is not a pure *here-and-now* will be expressed in general as a policy tree.

In the second step we solve the stochastic CSP under the given policy by finding specific *policy parameters*. In a one-stage stochastic CSP, the decision variables are set before the stochastic variables and the chosen policy is *here-and-now*. Under any other policy, that is *wait-and-see* or hybrid, we have an m-stage stochastic CSP where V and S are partitioned into disjoint sets, $V_1, \ldots, V_m$ and $S_1, \ldots, S_m$. To solve an m-stage stochastic CSP an assignment to the variables in V_1 must be found such that, given random values for S_1, an assignment can be found for V_2 such that, given random values for $S_2 \ldots$, an assignment can be found for V_m so that, given random values for S_m the hard constraints are satisfied and the chance-constraints are satisfied in the specified fraction of all possible scenarios.

In [98] a policy based view of stochastic constraint programs is proposed. The semantics is based on a tree of decisions. Each path in a policy represents a different possible scenario (set of values for the stochastic variables), and the values assigned to decision variables in this scenario. To find satisfying policies, backtracking and forward checking algorithms, which explores the implicit AND/OR graph, are presented. Such an approach has been further investigated in [5]. An alternative semantics for stochastic constraint programs, which suggests an alternative solution method, comes from a scenario-based view [11]. In [91] the authors outline this solution method, which consists in generating a scenario-tree that incorporates all possible realizations of discrete random variables into the

model explicitly. The great advantage of such an approach is that conventional constraint solvers can be used to solve stochastic CSP. Of course, there is a price to pay in this approach, as the number of scenarios grows exponentially with the number of stages and such a growth is particularly affected by random variables that contain a wide range of values in their domain. To deal with this problem the authors developed dedicated scenario-reduction techniques, which unfortunately affect the completeness of the approach when applied to improve performances of the search process. Another limit of the approaches in [98] and [91] is that they provide implementations only for a *wait-and-see* policy. The reason for this is that, when decision and random variables are split into disjoint sets $V_1, \ldots, V_m$ and $S_1, \ldots, S_m$ containing more than one element, the computation required to find policy parameters usually is special purpose and it is unlikely to be performed by a general approach.

2.2.4 Inventory control and (R^n,S^n) policy

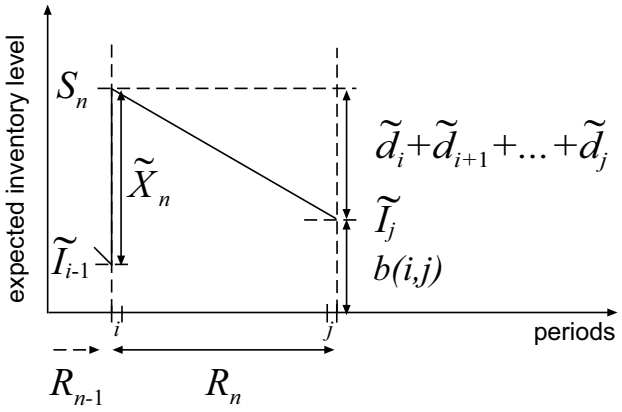


Figure 2.1: (R^n,S^n) policy. $\tilde{d}_i + \tilde{d}_{i+1} + \ldots + \tilde{d}_j$ is the expected demand over R^n; $b(i,j)$ is the minimum buffer stock required to guarantee service level α; $\tilde{X}_n$ is the expected order quantity in period i for replenishment cycle n; $\tilde{I}_{i-1}$ and $\tilde{I}_j$ are respectively the expected closing-inventory-levels for periods $i-1$ and j.

In this paper we consider the class of production/inventory control problems that refers to the single location, single product case under non-stationary stochastic demand. We consider the following inputs: a planning horizon of N periods and a demand d_t for each period $t \in \{1, \ldots, N\}$, which is a random variable

with probability density function $g_t(d_t)$. In the following sections we will assume, without loss of generality, that these variables are normally distributed. We assume that the demand occurs instantaneously at the beginning of each time period. The demand we consider is non-stationary, that is it can vary from period to period, and we also assume that demands in different periods are independent. A fixed delivery cost a is considered for each order and also a linear holding cost h is considered for each unit of product carried in stock from one period to the next.

We assume that it is not possible to sell back excess items to the vendor at the end of a period. As a service level constraint we require the probability that at the end of every period the net inventory will not be negative to be at least a given value α. Our aim is to find a replenishment plan that minimizes the expected total cost, which is composed of ordering costs and holding costs, over the N-period planning horizon, satisfying the service level constraints.

Different inventory control policies can be adopted for the described problem. A policy states the rules to decide when orders have to be placed and how to compute the replenishment lot-size for each order. For a discussion of inventory control policies see [81]. In what follows the problem described above will be solved adopting the replenishment cycle policy (R^n,S^n). We recall that R^n denotes the length of the nth replenishment cycle and S^n the respective order-up-to-level (Fig. 2.1). In this policy the actual order quantity X_n for replenishment cycle n is determined only after the demand in former periods has been realized. X_n is computed as the amount of stock required to raise the closing inventory level of replenishment cycle $n-1$ up to level S^n. In order to provide a solution for our problem under the (R^n, S^n) policy we must populate both the sets R^n and S^n for $n = \{1, \ldots, N\}$.

2.3 Existing approaches

Early works in stochastic inventory control area adopted heuristic strategies such as those proposed by Silver [80], Askin [3] and Bookbinder & Tan [15]. The first complete (MIP) solution method, which operates under mild assumptions,

was introduced for this problem by Tarim & Kingsman [89]. Tarim & Smith [92] introduced a more compact and efficient CP formulation for the same model. Dedicated *cost-based filtering* techniques for such a CP model were presented in [87] and [88]. This latter enhanced model proved to be able to solve real world problem instances considering up to a 50 periods planning horizon in a few seconds. In the following sections we discuss the assumptions adopted by Tarim & Kingsman and we propose a stochastic constraint programming approach in which these assumptions are dropped. By means of this approach we can compute optimal (R^n, S^n) policy parameters and the real associated expected total cost. Of course there is a price to pay for dropping Tarim & Kingsman's assumptions, in fact our approach is less efficient than the one proposed in [88].

2.3.1 Stochastic programming model

The stochastic programming formulation for the general multi-period production/inventory problem with stochastic demand can be expressed as finding the timing of the stock reviews and the size of the non-negative replenishment orders, X_t in period t, with the objective of minimizing the expected total cost $E\{TC\}$ over a finite planning horizon of N periods. The model is given below:

$$\begin{aligned} \min \; E\{TC\} = \int_{d_1}\int_{d_2}\dots\int_{d_N} \sum_{t=1}^{N} (a\delta_t + h \cdot \max(I_t, 0)) \\ g_1(d_1)g_2(d_2)\dots g_N(d_N)\mathrm{d}(d_1)\mathrm{d}(d_2)\dots\mathrm{d}(d_N) \end{aligned} \tag{2.3}$$

subject to, for $t = 1 \dots N$

$$\delta_t = \begin{cases} 1, & \text{if} \quad X_t > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

$$I_t = I_0 + \sum_{i=1}^{t}(X_i - d_i) \tag{2.5}$$

$$\Pr\{I_t \geq 0\} \geq \alpha \tag{2.6}$$

$$I_t \in \mathbb{R}, \quad X_t \geq 0, \quad \delta_t \in \{0, 1\}. \tag{2.7}$$

The demand d_t in each period is a continuous random variable with probability distribution function $g_t(d_t)$. Each decision variable I_t represents the inventory level at the end of period t. The binary decision variables δ_t state whether a replenishment is fixed for period t ($\delta_t = 1$) or not ($\delta_t = 0$). Chance-constraint (2.6) enforces the required service level, that is the probability α the net inventory will not be negative at the end of each and every time period. The objective function (2.3) minimizes the expected total cost over the given planning horizon.

Although this stochastic programming approach fully models our production/inventory problem, a solution cannot be expressed before a *response policy* is chosen. We have already seen that a policy states the rules to decide when decision variables have to be set. By using the general approach proposed in [91] a solution can be found under *wait-and-see* policy. In this policy a replenishment decision X_k for period k is made only after all the outcomes for random variables associated with former periods $1, \ldots, k-1$ have been observed. The solution therefore is expressed as a policy tree, which can exponentially grow in dimension even for short planning horizons.

In order to avoid this intractable solution, approaches based on order-up-to-level strategies have typically been proposed for this model in the literature. Expressing replenishment decisions in terms of order-up-to-levels instead of order quantities is a convenient way to find optimal policy parameters without employing an exponential solution tree. An order-up-to-level for period k represents the level to which stocks have to be maintained at the beginning of such a period. Therefore at the beginning of each period k, $k = 1 \ldots, N$, in our planning horizon we can observe the actual inventory level and we can decide if an order has to be issued to bring the inventory up to the required level. There are two well-known order-up-to-level policies for the general model proposed.

The so-called (s^n,S^n) policy [81] is a pure *wait-and-see* policy where at the end of period k we observe the inventory level and if this level is below s^k, then an order is issued to raise stocks up to level S^k. It is easy to see that this policy is *wait-and-see* since every decision, placing or not an order and the actual size of the order, is taken at the very last moment, by observing the demands that have been realized in the former periods. Furthermore a solution under this policy can be expressed by using only N pairs (s^k,S^k), in contrast to the exponential solution

tree required when the problem is modeled using order quantities.

A hybrid order-up-to-level policy is the so-called (R^n,S^n) policy [15], also known as replenishment cycle policy, which we described above. In this policy the inventory review times are set under a *here-and-now* strategy at the beginning of the planning horizon. These decisions are not affected by the actual demand realized in each period. On the other hand, for each inventory review we need to observe the actual demand realized in former periods to compute the actual order quantity. This makes the (R^n,S^n) policy hybrid, since the order quantity for each review is computed in a *wait-and-see* fashion only after previous demands have been realized. Also in this case the solution can be efficiently expressed. In fact we only require M ($\leq N$) couples of values (R^k,S^k), $k = 1, \ldots, M$, where R^k is the length of the k-th replenishment cycle and S^k is the respective order-up-to-level.

From these considerations, and from the well known Jensen's inequality [11], it is easy to see that an (s^n,S^n) policy always has a lower expected total cost than an (R^n,S^n) policy. The optimality of the (s^n,S^n) policy has been presented in [76]. In what follows we will focus on the (R^n,S^n) policy. In fact, as already discussed, despite being suboptimal this policy presents several interesting aspects.

In the next section we will recall a CP model proposed by Tarim and Smith [92] and based on a *deterministic equivalent* mathematical programming (MIP) model originally introduced by Tarim & Kingsman in [89] to compute (R^n,S^n) policy parameters. This model can only provide near-optimal policy parameters because it relies on assumptions that affect optimality. In the following section these assumptions are discussed.

2.3.2 Tarim & Kingsman's approach

In this section we provide a description of the *deterministic equivalent* CP formulation for the (R^n,S^n) policy proposed by Tarim and Smith in [92] and based on the approach originally introduced by Tarim and Kingsman in [89]. It should be noted that this formulation is the discrete version of the model presented in Section 2.3.1. Since the normal distribution is the limiting case of a discrete bi-

nomial distribution $P_p(k|n)$[†] as the sample size n becomes large[‡], in the discrete model an uniformly distributed random demand with mean μ and variance σ^2 can be modeled as a discrete random variable following a binomial probability mass function $P_p(k|n)$, where $np = \mu$ and $np(1-p) = \sigma^2$.

The *deterministic equivalent* CP formulation for the (R^n,S^n) policy proposed in [92] is

$$\min \; E\{TC\} = \sum_{t=1}^{N} \left(a\delta_t + h\tilde{I}_t \right) \tag{2.8}$$

subject to, for $t = 1 \ldots N$

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} \geq 0 \tag{2.9}$$

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} > 0 \Rightarrow \delta_t = 1 \tag{2.10}$$

$$\tilde{I}_t \geq b \left(\max_{j \in \{1..t\}} j \cdot \delta_j, t \right) \tag{2.11}$$

$$\tilde{I}_t \in \mathbb{Z}^+ \cup \{0\}, \quad \delta_t \in \{0, 1\} \tag{2.12}$$

where $b(i, j)$ is defined by

$$b(i,j) = G^{-1}_{d_i + d_{i+1} + \ldots + d_j}(\alpha) - \sum_{k=i}^{j} \tilde{d}_k. \tag{2.13}$$

$G_{d_i+d_{i+1}+\ldots+d_j}$ is the cumulative probability distribution function of $d_i + d_{i+1} + \ldots + d_j$. It is assumed that G is strictly increasing, hence G^{-1} is uniquely defined. Unfortunately the computation of the binomial cumulative distribution function is time consuming. For this reason it is common to adopt an approximate approach that exploits the respective normal cumulative distribution function[§], whose computation is much easier. In what follows we will adopt this approach not only for its efficiency, but also because it lets us comply in the discrete model with the

[†]The binomial distribution gives the discrete probability distribution $P_p(k|n)$ of obtaining exactly k successes out of n Bernoulli trials [52]

[‡]In which case $P_p(k|n)$ is normal with mean $\mu = np$ and variance $\sigma^2 = np(1-p)$.

[§]This approximation is a huge time-saver (exact calculations of $P_p(k|n)$ with large n are very onerous); it can be seen as a consequence of the central limit theorem [52] since $P_p(k|n)$ is a sum of n independent, identically distributed 0-1 indicator variables.

original problem definition that assumes a normally distributed demand in each period. We will therefore compute buffer stock levels as

$$b(i,j) = \text{round}\left(G^{-1}_{d_i,d_{i+1},\ldots,d_j}(\alpha)\right) - \sum_{k=i}^{j} \tilde{d}_k,$$

where $d_i, d_{i+1}, \ldots, d_j$ are normally distributed random variables. The term $G^{-1}_{d_i+d_{i+1}+\ldots+d_j}(\alpha)$ is rounded to the nearest integer — function $\text{round}(\cdot)$ — according to the known concept of *continuity correction* (see [24]) in probability theory. For a detailed discussion on this CP model see [87]. Each decision variable $\tilde{I}_t$ represents the expected inventory level at the end of period t. It should be noted that the expected inventory level at the beginning of such a period is simply $\tilde{I}_t + \tilde{d}_t$ and if a replenishment is scheduled in t this latter value denotes the order-up-to-level (S^n) in period t. Each $\tilde{d}_t$ represents the expected demand in a given period t according to its probability mass function $g_t(d_t)$. The binary decision variables δ_t state whether a replenishment is fixed for period t ($\delta_t = 1$) or not ($\delta_t = 0$). The objective function (2.8) minimizes the expected total cost over the given planning horizon. The two terms that contribute to the expected total cost are ordering costs and inventory holding costs. Constraint (2.9) enforces a no-buy-back condition, which means that received goods cannot be returned to the supplier. As a consequence of this the expected inventory level at the end of period t must be no less than the expected inventory level at the end of period $t-1$ minus the expected demand in period t. Constraint (2.10) expresses the replenishment condition. We have a replenishment if the expected inventory level at the end of period t is greater than the expected inventory level at the end of period $t-1$ minus the expected demand in period t. This means that we received some extra goods as a consequence of an order. Constraint (2.11) enforces the required service level α. This is done by specifying the minimum buffer stock required for each period t in order to assure that, at the end of every time period, the probability that the net inventory will not be negative is at least α. These buffer stocks, which are stored in matrix $b(\cdot,\cdot)$, are pre-computed following the approach originally suggested in [89].

The CP formulation operates under the assumption that negative orders are not

allowed, so that if the actual stock exceeds the order-up-to-level for that review, this excess stock is carried forward and not returned to the supply source. However this event is assumed to be rare, therefore in the model it is ignored (Fig. 2.2).

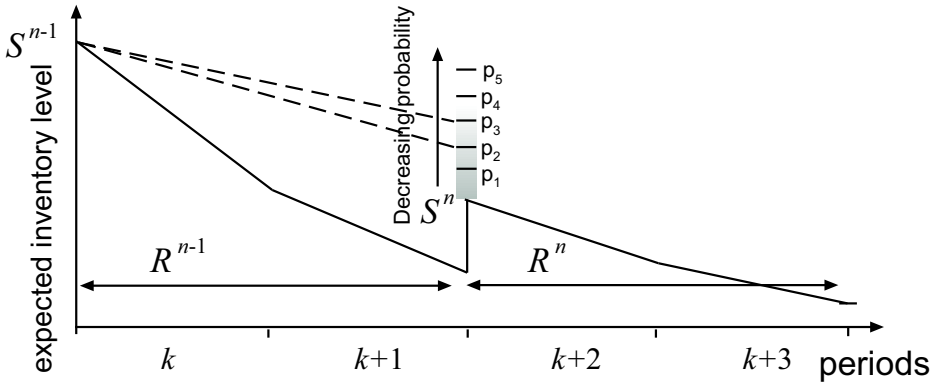


Figure 2.2: In Tarim & Kingsman [89] the event that actual stock exceeds the order-up-to-level S^n for a given review R^n is assumed to be rare. In other words, in their model observing a low demand during R^{n-1} has negligible probability. This implies that probabilities $p_1, p_2, \ldots, p_m$ are assumed to be low.

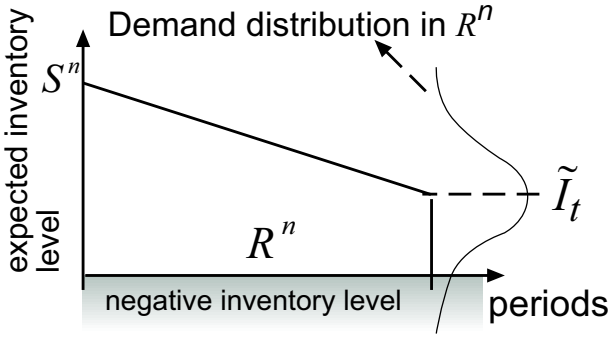


Figure 2.3: Negative inventory levels.

Let us analyze the effects of this assumption on the solutions produced by the CP approach.

1. The cost of carrying excess stock as a consequence of a low demand before a given replenishment is ignored, therefore the actual cost of a policy can be higher than the one provided by the model.

2. The event of carrying excess stock as a consequence of low demand before a given replenishment can have an impact on the service level of next periods. In particular, when the probability of ending up with a stock level higher than the order-up-to-level fixed in a given replenishment period is sufficiently high, it could be possible to exploit excess stock to provide the required service level, keeping lower expected closing inventory levels in following periods.

Furthermore, the CP approach models holding cost by considering expected closing-inventory-level values $\tilde{I}_t$ in each period (Fig. 2.3), while in the original stochastic programming formulation negative inventories do not contribute to the actual overall expected holding cost, which may be therefore higher than the one computed by the CP model.

2.4 A stochastic constraint programming approach based on global chance-constraints

In this section we provide a novel CP approach to find optimal (R^n, S^n) policy parameters. Our approach avoids both the assumptions adopted in Tarim and Kingsman [89], therefore it considers the effect of excess stock on the service level of subsequent replenishment cycles and on the expected total cost of a given policy. It also considers the fact that a negative closing-inventory-level does not contribute to the overall holding cost. The core of our modeling strategy is the new concept of *global chance-constraints*. By means of this novelty we are able to dynamically compute the exact service level provided by a given policy parameter configuration and the expected total cost associated with it.

2.4.1 Chance-constraints and policies

The techniques proposed in [98] and [91] for solving stochastic CSPs are general-purpose but limited to *wait-and-see* policies. Since in the inventory control problem presented we apply a hybrid policy, we adopt a different and specialized approach. By recalling that we can define a constraint as a mathematical function, in a similar fashion it is possible to define a *chance-constraint*, originally intro-

duced by Charnes and Cooper [18], as a mathematical function. Depending on the chosen policy the domain of our function f will change. For instance if we restrict ourselves to a *here-and-now* policy, so that the solution for our stochastic CSP can be expressed as a simple assignment for the decision variables, the function will be $f : D(x_1) \times \ldots \times D(x_n) \rightarrow \{0, 1\}$, where $V = \{x_1, \ldots, x_n\}$, and $f(x_1, \ldots, x_n) = 1$ if and only if $x_1, \ldots, x_n$ is an assignment such that, given random values for $y_1, \ldots, y_n$, where $S = \{y_1, \ldots, y_n\}$ the hard constraints are satisfied and the chance-constraints are satisfied in the specified fraction of all possible scenarios. In a *wait-and-see* policy as we have seen $V_1 = \{x_1\}, \ldots, V_n = \{x_n\}$ and $S_1 = \{y_1\}, \ldots, S_n = \{y_n\}$. Therefore the function $f(x_1, x_2, \ldots, x_n)$ will map each possible *policy tree* in the solution space identified by our chance-constraint to the two possible values $\{0, 1\}$. $f(x_1, x_2, \ldots, x_n) = 1$ if and only if the assignment for the variable x_1 is such that, given a random value for y_1, an assignment can be found for variable x_2 such that, given a random value for $y_2 \ldots$, an assignment can be found for variable x_m so that, given a random value for y_m the hard constraints are satisfied and the chance-constraints are satisfied in the specified fraction of all possible scenarios. These functions can obviously be expressed in theory for any possible policy.

2.4.2 Global chance-constraints

We recalled a known concept in stochastic programming: chance-constraints. We also saw in former sections how CP can be extended to consider random variables and chance-constraints. This leads to what is called *stochastic constraint programming*. We now aim to extend stochastic constraint programming with a new concept in analogy to what has been done for CP. We already saw in Section 2.2 that in CP the simultaneous presence of several simple constraints, for efficiency and expressiveness, is typically modeled by means of *global constraints*. Also in *stochastic programming* we can identify simple chance-constraints of the form $\Pr\{D \geq r\} \geq \alpha$, typically involving a decision variable D and a random variable r. An example is given by the service level at period t in our inventory control problem, $\Pr\{I_t \geq 0\} \geq \alpha$. These simple chance-constraints in stochastic programming typically appear as a set. In our inventory model we enforce a

service level constraint for every period in our planning horizon, that is we replicate $\Pr\{I_t \geq 0\} \geq \alpha$, for $t = 1, \ldots, N$. In a *stochastic constraint programming* framework it is therefore natural to group this set of simple chance-constraints and to define what we will call a *global chance-constraint* over a set of decision variables and a set of random variables. The general signature for a global chance-constraint will be

$$globalChanceConstraint(D_1, \ldots, D_N, r_1, \ldots, r_N, \alpha),$$

where $D_1, \ldots, D_N$ are decision variables $r_1, \ldots, r_N$ are random variables and α is a value in the interval $[0, 1]$, indicating the minimum satisfaction probability for the chance-constraint. According to the probability distribution functions of random variables, the filtering algorithm of this constraint will prune values from domains of $D_1, \ldots, D_N$ that cannot guarantee the chance-constraints are satisfied at the required threshold probability. Depending on the given problem and on the response policy chosen, dedicated efficient filtering algorithms can be implemented (see the forward checking technique proposed by Walsh [98] for *wait-and-see* policies, and the improved algorithm in [5]).

This new concept defines much more than a notation extension. In fact it should be noted that stochastic programming is a very high level modeling framework. An apparently simple constraint like the one presented, $\Pr\{I_t \geq 0\}$, actually hides in the stochastic programming model interdependencies between several, and often all, decision variables and random variables in the problem. Usually evaluating these dependencies requires the computation of a convolution integral. Therefore in general it will not be possible to express a global chance-constraint in stochastic constraint programming as a set of simple and independent chance-constraints. An immediate example is given by Tarim and Smith's model [92]. Here the *chance-constraints* in the stochastic programming model are modeled as independent deterministic equivalent constraints according to the approach proposed by Tarim and Kingsman [89]. As discussed in the former sections this leads to several approximations, since many dependencies between decision and random variables are ignored. In the following sections we introduce a global chance-constraint able to model these dependencies.

2.4.3 A global chance-constraint for (R^n,S^n) policy

We focus on the (R^n,S^n) policy, which is hybrid and therefore cannot be solved by means of the approaches in [5, 91] that only cope with wait-and-see policies. As already discussed, by reasoning in terms of order-up-to-levels, under this policy a solution for our stochastic model can be efficiently expressed as an assignment for our decision variables, that is replenishment decisions and order-up-to-levels, and it does not require a tree representation. We developed a dedicated *global chance-constraint* that identifies feasible policy parameters for our inventory control problem. As in the case of hard constraints the function does not necessarily have closed mathematical form. In our case this function is defined by providing an algorithm able to identify feasible assignments for decision variables, i.e. policy parameters. Within the same constraint we also developed an algorithm to compute the expected total cost for a given policy parameter configuration. The signature of our global chance-constraint is as follows

$$serviceLevelRS(C, a, h, \tilde{I}, \delta, d, \alpha)$$

where C is a decision variable denoting the expected total cost, a is the fixed ordering cost, h is the holding cost per unit, $\tilde{I}$ and δ are arrays of decision variables, d is an array of discrete random variables d_t with probability mass function $g_t(d_t)$ and α is the required service level. This constraint ensures that, at the end of each time period, the probability that the net inventory will not be negative is at least α. It is therefore semantically equivalent to Constraint (2.6) for $t = \{1, \ldots, N\}$ and it can be used to express these constraints in a CP model. The decision variable C represents a lower bound on the expected total cost (Eq. 2.3) for a given partial assignment for decision variables $\tilde{I}$ and δ, and such a bound is tight when all the decision variables $\tilde{I}$ and δ are ground. It should be noted that the *global view* provided by this constraint allows us to consider joint probabilities during the search when service levels and the expected total cost are computed. These joint probabilities are ignored when the same condition is expressed by means of many independent constraints as in Tarim and Smith [92]. In the following sections we will describe the deterministic equivalent CP model that incorporates our global chance-constraint and the propagation logic for the constraint.

2.4.4 Deterministic equivalent model

The deterministic equivalent model that incorporates our constraint is

$$\min \; E\{TC\} = C \tag{2.14}$$

subject to

$$serviceLevelRS(C, a, h, \tilde{I}_{t\in\{1,\ldots,N\}}, \delta_{t\in\{1,\ldots,N\}}, d_{t\in\{1,\ldots,N\}}, \alpha) \tag{2.15}$$

and for $t = 1 \ldots N$,

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} \geq 0 \tag{2.16}$$

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} > 0 \Rightarrow \delta_t = 1 \tag{2.17}$$

$$\tilde{I}_t, C \in \mathbb{Z}^+ \cup \{0\}, \quad \delta_t \in \{0, 1\}. \tag{2.18}$$

It is easy to see that the model is similar to the one proposed in [92] and presented in Section 2.3.2. Again we observe two sets of decision variables: the replenishment decision in period t, δ_t; and the expected closing-inventory-level in period t, $\tilde{I}_t$. The buffer stocks needed to provide the required service level α and the expected total cost C for a given policy are computed by the special purpose global chance-constraint.

2.4.5 Propagating the service level global chance-constraint

In order to propagate our constraint and compute a feasible assignment for the expected closing-inventory-levels $\tilde{I}$, we will consider now a two-replenishment cycle case (Fig. 2.4) in a four-period planning horizon, then we will extend the idea in a recursive fashion to the case of M subsequent replenishment cycles $\{R^1, \ldots, R^M\}$ over N periods. Two consecutive replenishment cycles are planned over the planning horizon considered, let us call them R^1 and R^2. R^1 covers periods $\{1, 2\}$, R^2 periods $\{3, 4\}$. Let S^i be the opening inventory level for R^i and $\Pr\{d_i \leq D\}$ be the probability of the event "observing a demand in period i less than or equal to D", where d_i is a random variable that represents the distri-

bution of the demand in period i. In a simple newsvendor problem [81] over one

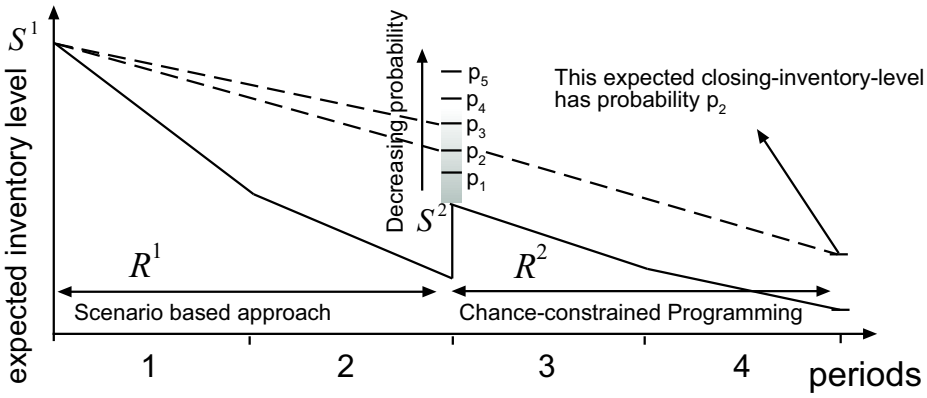


Figure 2.4: Two replenishment cycle case.

period with random demand d, the opening-inventory-level that provides a service level α can be computed as $G_d^{-1}(\alpha)$, where G_d^{-1} is the inverse cumulative distribution function of d. It is easy to see that $S^1 = G_{d_1+d_2}^{-1}(\alpha)$ and the correct minimum opening-inventory-level S^2 for R^2, which guarantees the required service level α, can be computed from the following relation that mixes *scenario-based approach* and *chance-constrained programming*

$$\begin{aligned} &\Pr\{d_1 + d_2 \geq S^1 - S^2\} \cdot G_{d_3+d_4}(S^2) + \\ &\sum_{i=0}^{S^1-S^2} \big(\Pr\{d_1 + d_2 = i\} \cdot G_{d_3+d_4}(S^1 - i)\big) \geq \alpha, \end{aligned} \tag{2.19}$$

where $G_{d_i+d_{i+1}+\ldots+d_j}(\cdot)$ is the cumulative probability distribution function of $d_i + d_{i+1} + \ldots + d_j$. For the two replenishment cycles case, this can be rewritten using the following extended form

$$\begin{aligned} &(1 - G_{d_1+d_2}(S^1 - S^2 - 1)) \cdot G_{d^3+d^4}(S^2) + \\ &\sum_{i=0}^{S^1-S^2} (G_{d_1+d_2}(i) - G_{d_1+d_2}(i-1)) \cdot G_{d_3+d_4}(S^1 - i) \geq \alpha. \end{aligned} \tag{2.20}$$

Notice that if S^1 is smaller than S^2, obviously the former cycle has no influence on the computation of S^2 and Condition 2.19 becomes $G_{d_3+d_4}(S^2) \geq \alpha$. Further-

more, if the computed S^2 is such that $S^2 < S^1 - \tilde{d}_1$, we just set S^2 to the minimum value allowed, that is $S^1 - \tilde{d}_1$.

Finally observe that the term

$$\sum_{i=1}^{S^1-S^2} (G_{d_1+d_2}(i) - G_{d_1+d_2}(i-1)) \cdot G_{d_3+d_4}(S^1 - i)$$

in Condition 2.20 has to be multiplied by the normalization term

$$G_{d_1+d_2}(S^1 - S^2 - 1) \Big/ \sum_{i=0}^{S^1-S^2} (G_{d_1+d_2}(i) - G_{d_1+d_2}(i-1))$$

in order to guarantee that the sum of all the event probabilities is one. In fact negative demands are disregarded, but the respective probabilities must be taken into account to cover the space of all possible events.

In order to propagate (Algorithm 1: `propagate`) this constraint in the case of M subsequent replenishment cycles over N periods, at each node of the search tree we look for the first M consecutive replenishment cycles (Algorithm 1, line 2) identified by the current partial assignment for decision variables δ. Two replenishment cycles R^m, R^{m+1} are consecutive if the last period of R^m is g and the first period of R^{m+1} is $g+1$. A replenishment cycle R^k over periods $\{i, \ldots, j\}$ can be identified by a full assignment over $\delta_i, \ldots, \delta_{j+1}$ where δ_i, δ_{j+1} are set to 1 and $\delta_{i+1}, \ldots, \delta_j$ are set to 0 (Function `listCycles()`). The opening-inventory-level S^1 for the first replenishment cycle R^1 covering periods $\{1, \ldots, j\}$ can be easily computed as $G^{-1}_{d_1+\ldots+d_j}(\alpha)$. In what follows we will describe a recursive *scenario-based approach* [11] to compute the opening-inventory-level S^j required in replenishment cycle $j \in \{1, \ldots, M\}$. We will assume that opening-inventory-levels for $R^1, \ldots, R^{j-1}$ are known (Algorithm 1, line 8) and we will use a generalized version of Condition 2.19 to compute such a value (Algorithm 1, lines 19 to 21). A generalized version of Eq. 2.19 for the case of M replenishment cycles can be introduced by observing that S^j, $j \in \{1, \ldots, M\}$, the opening-inventory-level for opening-inventory-level for replenishment cycle R^j, is affected only by former replenishment cycles $\{R^i, \ldots, R^{j-1}\}$, where $i = \min \{v \in \{1, \ldots, j\} | (S^v \geq S^1) \wedge \ldots \wedge (S^v \geq S^{v-1})\}$. If $i = j$ no former

replenishment cycle affects R^j. Now since we know the distribution of the demand in replenishment cycles $\{R^i, \ldots, R^j\}$ and under the assumption that former opening-inventory-levels $\{S^i, \ldots, S^{j-1}\}$ have been already set, it is easy to recursively compute the expected service level for replenishment cycle R^j by using a *scenario based approach*. We can therefore extend Condition 2.19 to compute S^j for R^j given that $\{R^i, \ldots, R^{j-1}\}$ are the former periods affecting service level of R^j.

Let $P_j(S^j)$ be the probability of observing an inventory level of S^j, that is the opening-inventory-level R^j, at the beginning of R^j.

Let $P_j(S^j, h)$ be the probability of observing an inventory level of $S^j + h$, that is h units higher than the opening-inventory-level of R^j, at the beginning of R^j.

Given $q \in \mathbb{Z}^+ \cup \{0\}$ and $k \in \{i, \ldots, M\}$, the probability associated with the event "observing a demand less or equal to q in replenishment cycle R^k" can be easily computed. Such a probability is in fact $G_{d_{R^k}}(q)$, where d_{R^k} is the demand distribution in replenishment cycle R^k, that is, if R^k covers periods $\{m, \ldots, n\}$, $d_{R^k} = d_m + \ldots + d_n$. Let $\widehat{G}_{d_{R^k}}(q)$ be the element of probability $G_{d_{R^k}}(q) - G_{d_{R^k}}(q-1)$.

- if $S^{j-1} \geq S^j$, then $P_j(S^j)$ is computed as

$$P_{j-1}(S^{j-1}) \cdot \left(1 - G_{d_{R^{j-1}}}(S^{j-1} - S^j - 1)\right) + \sum_{k=1}^{S^i - S^{j-1}} P_{j-1}(S^{j-1}, k) \cdot \left(1 - G_{d_{R^{j-1}}}(S^{j-1} - S^j + k - 1)\right) \tag{2.21}$$

  that is $P_{j-1}(S^{j-1})$ multiplied by the probability of the event "observing a demand greater or equal to $S^{j-1} - S^j$ in replenishment cycle R^{j-1}", plus the summation, for $k = 1, \ldots, S^i - S^{j-1}$, of $P_{j-1}(S^{j-1}, k)$ multiplied by the probability of the event "in R^{j-1} we observe a demand greater or equal to $S^{j-1} - S^j + k$".

- if $S^{j-1} < S^j$, then $P_j(S^j)$ is computed as

$$P_{j-1}(S^{j-1}) + \sum_{k=1}^{S^j - S^{j-1}} P_{j-1}(S^{j-1}, k) + \sum_{k=1}^{S^i - S^j} P_{j-1}(S^{j-1}, S^j - S^{j-1} + k) \cdot \left(1 - G_{d_{R^{j-1}}}(k-1)\right) \tag{2.22}$$

- if $S^{j-1} \geq S^j + h$, then $P_j(S^j, h)$ is computed as

$$P_{j-1}(S^{j-1}) \cdot \widehat{G}_{d_{R^{j-1}}}(S^{j-1} - S^j - h) + \sum_{k=1}^{S^i - S^{j-1} - h} P_{j-1}(S^{j-1}, k) \cdot \widehat{G}_{d_{R^{j-1}}}(S^{j-1} - S^j - h + k) \tag{2.23}$$

- if $S^{j-1} < S^j + h$, then $P_j(S^j, h)$ is computed as

$$\sum_{k=S^j+h-S^{j-1}}^{S^i - S^{j-1}} P_{j-1}(S^{j-1}, k) \cdot \widehat{G}_{d_{R^{j-1}}}(k - S^j - h + S^{j-1}). \tag{2.24}$$

Obviously $P_i(S^i) = 1$ since, for the way R^i is chosen, no former replenishment cycle may affect its order-up-to-level S^i. By following a *dynamic programming* [8] scheme, S^j can be computed as the minimum value that satisfies

$$P_j(S^j) \cdot G_{d_{R^j}}(S^j) + \sum_{k=1}^{S^i - S^j} \left(P_j(S^j, k) \cdot G_{d_{R^j}}(S^j + k)\right) \geq \alpha. \tag{2.25}$$

Since this paper is not focused on efficiency issues, the dynamic programming algorithm developed to implement Eq. 2.25 simply employs a recursive code structured as the functional equation itself. Nevertheless we want to underline that the proposed recursion only aims to describe a correct functional equation to compute feasible assignments. As in every dynamic program, efficiency can be obtained by adopting a forward recursion and by trading memory and time to avoid computing the probability of a given scenario more than once. In the recursive computation scenarios with negative demands are not considered, therefore we must normalize

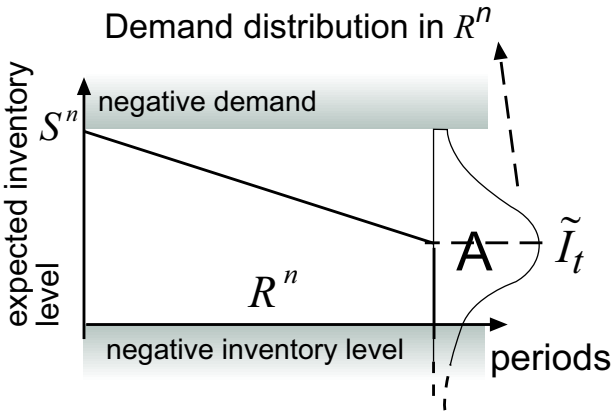


Figure 2.5: Normalization.

the probabilities of other events in order to ensure that their sum covers the whole space of the possible events. In other words we need to ensure that the probability associated with area A in Fig. 2.5 is one. This is a known approach in inventory control and it is usually justified since the distortion introduced by this normalization typically does not affect the quality of the solutions. A possible way to perform this normalization step is to divide the term

$$P_j(S^j) \cdot G_{d_{Rj}}(S^j) + \sum_{k=1}^{S^i - S^j} \left(P_j(S^j, k) \cdot G_{d_{Rj}}(S^j + k) \right)$$

in Condition 2.25 by the following normalization term

$$P_j(S^j) + \sum_{i=k}^{S^i - S^j} P_j(S^j, k) \tag{2.26}$$

in order to guarantee that the sum of all the probabilities of the events considered in step j is one.

In order to speed up the search for the optimal opening-inventory-level associated with a given replenishment cycle R^k, recall that opening-inventory-levels computed as shown in [92] are always greater than or equal to optimal opening-inventory-level satisfying Eq. 2.25. Therefore an efficient strategy (Procedure `setBufferForCycle()`) for finding optimal opening-inventory-levels is to

consider sequentially the first M replenishment cycles, R^k, $k \in \{1, \ldots, M\}$, identified by the current partial assignment for replenishment decisions δ. For each replenishment cycle R^k an upper-bound for the optimal opening-inventory-level can be computed as $\lceil G^{-1}_{d_{R^k}}(\alpha) \rceil$ (see [89]). Starting from this upper-bound we can decrease it and search for the minimum value that satisfies Eq. 2.25 (Procedure `setBufferForCycle()`, line 4). Opening-inventory-levels computed as in [89] are close to optimal because probabilities associated with negative order quantity scenarios are typically low, therefore this strategy requires only a few steps to reach the optimum levels.

2.4.6 Computing holding cost

In this section we address the problem of computing the correct holding cost for a given replenishment cycle R covering periods $\{i, \ldots, j\}$ when the expected closing-inventory-level $\tilde{I}_t$ for each period $t \in \{i, \ldots, j\}$ is given. We recall that $\tilde{I}_j$ denotes S^j minus the expected demand in replenishment cycle j, $\tilde{d}_{R^j}$. The problem of computing the exact holding cost arises from the fact that negative inventory levels do not contribute to the overall holding cost. Therefore the term $h\tilde{I}_t$ in the objective function of the model presented by Tarim & Kingsman is not a complete representation of this cost component. Once $\tilde{I}_j$ is known every other $\tilde{I}_k$, $k \in \{i, \ldots, j-1\}$ can be easily computed as $\tilde{I}_k = \tilde{I}_j + \sum_{t=k+1}^{j} \tilde{d}_t$. Let $h(R, \tilde{I}_j)$ be the expected holding cost for replenishment cycle R when the expected closing-inventory-level $\tilde{I}_j$ is given. This cost component is made up of individual cost components for each period in our replenishment cycle R. Let us consider a given period $k \in \{i, \ldots, j\}$. The opening inventory level for R is $S^i = \tilde{I}_j + \sum_{t=i}^{j} \tilde{d}_t$. We recall that the probability of observing an overall demand r over the time span $\{i, \ldots, k\}$ is denoted by $\widehat{G}_{d_i+\ldots+d_k}(r)$. By letting r range from 0 to S^i we obtain every possible scenario for which a holding cost is incurred in period k. Therefore the expected holding cost for period k can be expressed as $h \sum_{r=0}^{S^i} (S_i - r) \cdot \widehat{G}_{d_i+\ldots+d_k}(r)$ and the expected holding cost for replenishment cycle R will be the sum of the contributions from every period $k \in \{i, \ldots, j\}$.

Algorithm 1: `propagate`

input : C, $\delta_1, \ldots, \delta_N$, $\tilde{I}_1, \ldots, \tilde{I}_N$, α, a, h, $d_1, \ldots, d_N$, N

begin
  $cycles \leftarrow listCycles(\delta_1, \ldots, \delta_N, \tilde{I}_1, \ldots, \tilde{I}_N, N)$;
  $n \leftarrow$ # elements in $cycles$;
  if $n = 0$ **then**
    return;
  $cost \leftarrow a \cdot n$;
  $condition \leftarrow true$;
  for *each element* e *in* $cycles$ **do**
    let $\{i, \ldots, j\}$ be the span covered by e;
    if *no decision variable* $\tilde{I}_i, \ldots, \tilde{I}_j$ *is assigned* **then**
      $condition \leftarrow false$;
    else if $\exists k \mid$ *decision variable* $\tilde{I}_k$, $i \leq k \leq j$ *is assigned* **then**
      $S^i \leftarrow$ cycle opening inventory level of e, linearly dependent on $\tilde{I}_k$;
      $holdingCost \leftarrow$ cycle holding cost of e with opening inventory level S^i (Eq. 2.27);
      $cost \leftarrow cost + holdingCost$;
  if $condition$ **then**
    $C \leftarrow cost$;
  else
    $setBufferForCycle(cycles, d_1, \ldots, d_N, \alpha)$;
    let e be the last element in $cycles$, a replenishment cycle over $\{i, \ldots, j\}$;
    $S^i \leftarrow$ cycle opening inventory level of e, linearly dependent on $\tilde{I}_j$;
    $holdingCost \leftarrow$ cycle holding cost of e with opening inventory level S^i (Eq. 2.27);
    $cost \leftarrow cost + holdingCost$;
    $Inf(C) \leftarrow cost$;
end

Procedure `setBufferForCycle`($cycles, d_1, \ldots, d_N, \alpha$)

input: $cycles, d_1, \ldots, d_N, \alpha$

begin
- let R be the last element in $cycles$, a replenishment cycle over $\{i, \ldots, j\}$;
- $S \leftarrow \lceil G^{-1}_{d_i+\ldots+d_j}(\alpha) \rceil$;
- decrease S to the min value that satisfies Eq. 2.25, with former cycles as listed in $cycles$;
- $\tilde{I}_j \leftarrow x - \tilde{d}_i - ... - \tilde{d}_j$;

end

Function `listCycles`($\delta_1, \ldots, \delta_N, \tilde{I}_1 \ldots, \tilde{I}_N, N$)

input : $\delta_1, \ldots, \delta_N, N$

output: $cycles$

begin
- $cycles \leftarrow \{\}$;
- $lastCycle \leftarrow null$;
- $pointer \leftarrow 1$;
- **for** *each* δ_i, $i = 2, \ldots, N$ **do**
  - **if** δ_i *is not assigned* **then**
    - return $cycles$;
  - **else if** $lastCycle \neq null$ **then**
    - let $\{i, \ldots, j\}$ be the span covered by $lastCycle$;
    - **if** *no variable* $\tilde{I}_i, \ldots, \tilde{I}_j$ *is assigned* **then**
      - return $cycles$;
  - **if** δ_i *is assigned to* 1 **then**
    - $lastCycle \leftarrow$ a replenishment cycle over $\{pointer, ..., i-1\}$;
    - add $lastCycle$ to $cycles$;
    - $pointer \leftarrow i$;
- $lastCycle \leftarrow$ a replenishment cycle over $\{pointer, ..., N\}$;
- add $lastCycle$ to $cycles$;
- return $cycles$;

end

2.4.7 Computing the objective function

In order to compute the expected total cost for a given replenishment plan, or a lower bound for such a cost associated with a given partial assignment for replenishment decisions δ, we look again for the first M consecutive replenishment cycles identified by the current partial assignment for decision variables δ. Therefore we will assume that $R^1, \ldots, R^M$ are known (Algorithm 1, line 8) and we will follow a reasoning similar to the one developed to satisfy our chance-constraints.

The expected holding cost for replenishment cycle R^j, $j \in \{1, \ldots, M\}$, is affected only by former replenishment cycles $\{R^i, \ldots, R^{j-1}\}$, where $i = \min \{v \in \{1, \ldots, j\} | (S^v \geq S^1) \wedge \ldots \wedge (S^v \geq S^{v-1})\}$. If $i = j$ no former replenishment cycle affects R^j. Now since we know the distribution of the demand in replenishment cycles $\{R^i, \ldots, R^j\}$ and since we assume that former opening-inventory-levels $\{S^i, \ldots, S^{j-1}\}$ have been already set, it is easy to recursively compute the expected holding cost for replenishment cycle R^j by using a *scenario based approach*.

The expected holding cost (HC) for R^j given that $\{R^i, \ldots, R^{j-1}\}$ are the earlier periods affecting R^j can be computed as

$$E\{HC_{R^j}\} = P_j(S^j) \cdot h(R^j, \tilde{I}_j) + \sum_{k=1}^{S^i - S^j} \Big(P_j(S^j, k) \cdot h(R^j, \tilde{I}_j + i) \Big) . \quad (2.27)$$

Also in this case, since negative demands are not considered in the summation, event probabilities must be normalized accordingly using the term given in Eq. 2.26 as shown before.

A valid lower bound (Algorithm 1, line 24) for the expected total cost of a given partial assignment involving decision variables δ — tight when the assignment is complete (Algorithm 1, line 17) — can be computed by considering a fixed ordering cost for each replenishment cycle R^i identified by the assignment (Algorithm 1, line 6), plus the expected holding cost for the first M consecutive replenishment cycles $R^1, \ldots, R^M$ computed as explained above (Algorithm 1, lines 14 and 22).

2.4.8 Cost-based filtering

In order to improve the search process we employed a cost-based filtering method similar to the one proposed in [87]. We will not describe in detail the whole method. We will rather try to give a high level description of it. The reader may refer to [87] for further details.

Firstly we recall that, in Tarim and Kingsman's model [89], upper bounds for decision variables $\tilde{I}_i$, $i = \{1, \ldots, N\}$ can be computed by considering a single replenishment cycle covering the whole planning horizon. The buffer stock required to guarantee the required service level is $b(1, N)$, as defined in Eq. 2.13. Since $b(i, j)$ is an increasing function [92], it directly follows that the maximum value for the domain of $\tilde{I}_N$ is obviously $b(1, N)$ and that for every other decision variable $\tilde{I}_i$, $i = \{1, \ldots, N-1\}$ the maximum value in the domain is $b(1, N) + \sum_{k=i+1}^{N} \tilde{d}_k$. These bounds are still valid in our model. In fact the effect of excess stocks from former periods may only decrease a buffer stock needed to provide a given service level.

A lower bound for the cost of an optimal policy associated with a given partial assignment can be computed as shown in [87]. In this work the authors solve in polynomial time, by using a shortest path algorithm, a relaxation of the original problem where inventory conservation constraints between subsequent replenishment cycles are relaxed. This means that negative order quantities are allowed in this relaxed model. The bound is dynamically computed during the search process and it takes into account partial assignments for both decision variables δ_t and inventory levels $\tilde{I}_t$, by respectively forbidding or forcing stated nodes in the optimal path to reflect assignments for δ_t variables, and by modifying costs in the connection matrix to reflect assignments for $\tilde{I}_t$ variables.

A similar approach can be adopted in our case by noticing that Tarim and Kingsman's approach underestimates holding cost in each period. Firstly because it considers the contribution of negative inventory levels on the holding cost. Secondly because it does not consider the effect of excess stocks from former periods not only in the service level computation, but also in the cost computation. This means that Tarim and Kingsman's model always computes a cost that is less than or equal to the actual cost associated with a given policy. On the other hand, as

seen, such a model overestimates buffer stocks.

In our cost-based filtering approach we relax not only the inventory conservation constraints, as in [87], but also the constraints that force buffer stocks at the end of each replenishment cycle. Therefore we simply solve a deterministic production planning problem under fixed ordering cost and linear holding cost. The same algorithm proposed in [87] can be employed to efficiently solve this problem. Since we do not take into account buffer stocks, and from the former considerations on the cost structure, this relaxed Tarim and Kingsman model provides a lower bound for the cost provided by our exact model. Also in our cost-based filtering approach this bound is dynamically computed during the search process and it takes into account partial assignments for both decision variables δ_t and inventory levels $\tilde{I}_t$ as discussed above.

2.5 Comparison with Tarim & Kingsman's approach

In this section we compare the results obtained by the approach presented in [87] with the exact solutions provided by the new model.

The following assumptions are valid for the rest of this section. We assume that the demand in each period is normally distributed about the forecast value with the same coefficient of variation τ. Thus the standard deviation of demand in period t is $\sigma_t = \tau \cdot \tilde{d}_t$. In all cases, initial inventory levels, delivery lead-times and salvage values are set to zero.

All experiments here presented were performed on an Intel(R) Centrino(TM) CPU 1.50GHz with 500Mb RAM. The solver used for our test is Choco [58], an open-source solver developed in Java.

Firstly we consider a decreasing demand pattern over a 5-period planning horizon. The planning horizon considered is short since this demand pattern is particularly hard to treat.

The forecasts for the demand in each period are given in Table 2.1. As input parameters we considered $a \in \{1, 100, 200\}$, $\tau \in \{0.15, 0.25\}$ and $\alpha \in \{0.95, 0.75\}$. The holding cost h is fixed and equal to 1 for all the instances, since replenishment decisions are affected only by the ratio between ordering cost and holding cost. In Table 2.2 experimental results are presented. For each instance

Decreasing	Period	1	2	3	4	5
	$\tilde{d}_t$	400	130	150	60	35

Table 2.1: Expected values for a decreasing demand pattern.

				Total Cost						
	parameters			T&K				Exact		
	a	τ	α	$E\{TC\}$	$\widehat{E}\{TC\}$	gap(%)	sec	$E\{TC\}$	gap(%)	sec
1	1	0.25	0.95	324	370	12.4	1	358	3.35	469
2	100	0.25	0.95	773	814	5.04	1	799	1.88	254
3	200	0.25	0.95	1152	1189	3.11	1	1176	1.11	165
4	1	0.15	0.95	197	205	3.90	1	200	2.50	372
5	100	0.15	0.95	637	644	1.09	1	640	0.63	249
6	200	0.15	0.95	984	990	0.61	1	985	0.51	30
7	1	0.25	0.75	135	178	24.1	1	172	3.49	219
8	100	0.25	0.75	573	613	6.53	1	607	0.99	161
9	200	0.25	0.75	886	910	2.64	1	907	0.33	22
10	1	0.15	0.75	83	101	17.8	1	100	1.00	282
11	100	0.15	0.75	517	535	3.36	1	534	0.19	181
12	200	0.15	0.75	797	810	1.60	1	809	0.12	8

Table 2.2: Decreasing demand pattern. Columns "$E\{TC\}$" are the expected total cost computed by Tarim and Kingsman's approximate approach (T&K) and by our exact approach (Exact). In order to compute T&K $E\{TC\}$ we employed the efficient CP approach proposed in [87]. In columns "sec" we report, in seconds, the time performance for each model. Since T&K provides an approximate expected total cost, in column "$\widehat{E}\{TC\}$" we report the actual expected total cost of such a solution, which is computed by simulating demands according to the given distribution in each period and by observing the realized total cost over 10000 runs. The two columns "gap" for T&K and Exact report respectively: the difference between T&K $E\{TC\}$ and T&K $\widehat{E}\{TC\}$, in percentage on T&K $E\{TC\}$, and the difference between T&K $\widehat{E}\{TC\}$ and Exact $E\{TC\}$ in percentage on Exact $E\{TC\}$. Holding cost h is set to 1 for every instance.

considered "Exact $E\{TC\}$" is the expected total cost of the optimal solution (i.e. set of policy parameters: replenishment cycle lengths and order-up-to-levels) obtained using the complete approach we presented. "T&K $E\{TC\}$" is the approximate expected total cost of the solution obtained by using the model proposed in [87], which adopts Tarim & Kingsman's approach. "T&K $\widehat{E}\{TC\}$" is the actual expected total cost of the solution obtained using the model proposed in [87]. This actual expected total cost has been computed by simulation. Notice that for some parameter configurations the solution obtained with the approach in [92] differs

from the optimal one, while for other cases the approximate approach produces a solution close to the optimal one. The reasons are different depending on the particular parameter configuration.

Instance (1) has a low ordering cost a, therefore we expect to order frequently. The expected total holding cost and the buffer stock levels required to provide service level α are affected by the negative trend of the demand and by excess stocks carried from former replenishment cycle as a consequence of this trend (Fig. 2.6). Since the model in [87] does not take into account these effects the

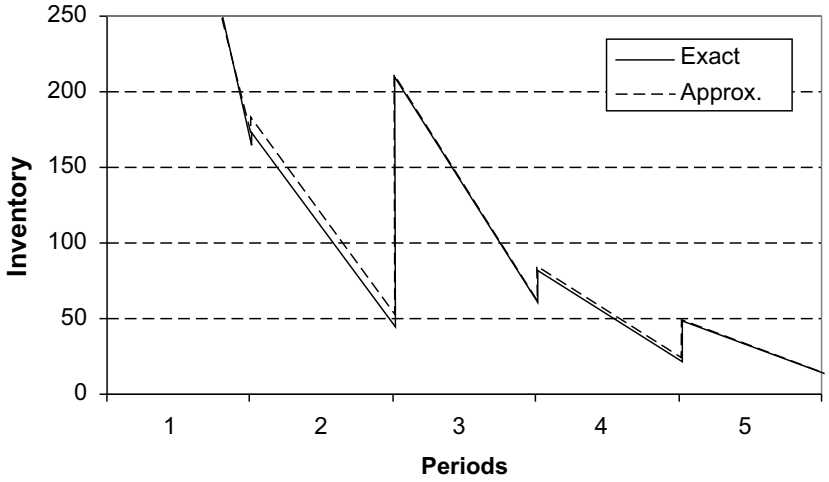


Figure 2.6: Comparison between inventory levels computed by the exact and the approximate approach.

expected total cost of the optimal solution it provides (T&K $\widehat{E}\{TC\}$) differs from the actual optimum (Exact $E\{TC\}$).

Instances (10), (11) and (12) have a low service level α and coefficient of variation τ. In this case the policy parameters computed by the approach in [87] are optimal, in fact T&K $\widehat{E}\{TC\}$ is close to Exact $E\{TC\}$. The effect of excess stocks is so low that it can actually be ignored, but the approximate expected total cost computed by the approach in [87] (T&K $E\{TC\}$) differs from the exact one (T&K $\widehat{E}\{TC\}$) by respectively 17.8%, 3.36% and 1.60%, since negative inventory levels affect the expected total cost of the policy. This follows from the fact that we require a low service level and we keep low buffer stock levels, therefore the probability of ending up with negative inventory levels becomes high and the

	Period	1	2	3	4	5	6	7	8
Seasonal	$\tilde{d}_t$	50	75	90	75	50	25	10	25
Life cycle	$\tilde{d}_t$	20	25	30	35	40	25	20	10
Erratic	$\tilde{d}_t$	50	30	70	15	60	10	30	15

Table 2.3: Expected values for Seasonal, Life Cycle and Erratic demand patterns.

effect of negative inventory levels on the expected holding cost increases as the length of the replenishment cycles decreases.

It should be noted that the computational effort required by our exact approach to compute policy parameters is directly affected by the number of replenishment cycles in our plan. This is the reason why we observe higher run times when the ratio between ordering cost and holding cost is low. This is true in general also for the instances that will be considered below.

We will now consider three other demand patterns that typically arise in practice. These patterns were originally proposed by Berry in [10] and they were also adopted for the experiments in [89]. The patterns are presented in Table 2.3. We did not consider a constant demand pattern, which is instead included in Berry's test bed, since it is obvious that for this pattern the solutions provided by our approach would not differ from the ones provided by Tarim's and Kingsman approach. In these cases as input parameters we considered $a \in \{1, 50, 100\}$, $\tau \in \{0.2, 0.3\}$ and $\alpha \in \{0.95, 0.75\}$. In Table 2.4 experimental results for these three further demand patterns are presented. Similar considerations to those just introduced indicate why also for these demand patterns in some cases the results provided by our exact approach may differ substantially from those obtained with the approximate one. Typically such a difference is due to the combined effect of excess stocks and/or negative inventory levels as already discussed.

From our experiments it is clear that the approximate expected total cost computed by Tarim & Kingsman's model (T&K $E\{TC\}$) may substantially underestimate the exact expected total cost (T&K $\widehat{E}\{TC\}$) associated with a given solution, which can be easily computed by simulation or by using our exact model. This is particularly evident in the erratic demand case, where for instances 43 and 46 the approximate expected total cost predicted by Tarim & Kingsman's model (T&K

	parameters			Total Cost						
				T&K				Exact		
	a	τ	α	$E\{TC\}$	$\widehat{E}\{TC\}$	gap(%)	sec	$E\{TC\}$	gap(%)	sec
13	1	0.3	0.95	205	213	3.76	1	207	2.90	2774
14	50	0.3	0.95	566	570	0.70	1	564	1.06	478
15	100	0.3	0.95	858	864	0.69	1	859	0.58	104
16	1	0.2	0.95	139	140	0.71	1	139	0.72	1412
17	50	0.2	0.95	498	499	0.20	1	498	0.20	180
18	100	0.2	0.95	771	772	0.13	1	766	0.78	66
19	1	0.3	0.75	88	108	18.5	1	106	1.89	908
20	50	0.3	0.75	440	458	3.93	1	458	0.00	165
21	100	0.3	0.75	696	710	1.97	1	709	0.14	56
22	1	0.2	0.75	61	73	16.4	1	72	1.39	603
23	50	0.2	0.75	411	422	2.61	1	420	0.48	109
24	100	0.2	0.75	658	666	1.20	1	665	0.15	51
25	1	0.3	0.95	109	110	0.91	1	110	0.00	48
26	50	0.3	0.95	441	443	0.45	1	438	1.14	8
27	100	0.3	0.95	634	634	0.00	1	630	0.63	4
28	1	0.2	0.95	76	77	1.30	1	77	0.00	34
29	50	0.2	0.95	393	393	0.00	1	392	0.26	6
30	100	0.2	0.95	574	574	0.00	1	570	0.70	4
31	1	0.3	0.75	49	58	15.5	1	56	3.57	30
32	50	0.3	0.75	355	362	1.93	1	357	1.40	6
33	100	0.3	0.75	529	535	1.12	1	531	0.75	4
34	1	0.2	0.75	35	41	14.6	1	40	2.50	27
35	50	0.2	0.75	333	338	1.48	1	334	1.20	6
36	100	0.2	0.75	503	507	0.79	1	503	0.80	4
37	1	0.3	0.95	175	195	10.2	1	188	3.72	554
38	50	0.3	0.95	492	494	0.40	1	489	1.02	33
39	100	0.3	0.95	692	692	0.00	1	689	0.44	14
40	1	0.2	0.95	110	122	9.84	1	119	2.52	381
41	50	0.2	0.95	418	418	0.00	1	417	0.24	25
42	100	0.2	0.95	618	619	0.16	1	617	0.32	10
43	1	0.3	0.75	64	90	28.8	1	85	5.88	277
44	50	0.3	0.75	360	370	2.70	1	369	0.27	18
45	100	0.3	0.75	560	570	1.75	1	569	0.18	9
46	1	0.2	0.75	45	59	23.7	1	56	5.36	225
47	50	0.2	0.75	332	339	2.06	1	339	0.00	19
48	100	0.2	0.75	532	539	1.30	1	536	0.56	8

Table 2.4: Experimental results for Seasonal ($13, \ldots, 24$), Life Cycle ($25, \ldots, 36$) and Erratic ($37, \ldots, 48$) demand patterns.

$E\{TC\}$) is respectively 28.8% and 23.7% less costly than the exact expected total cost associated with the policy parameter configuration in the respective solution (T&K $\widehat{E}\{TC\}$). Although Tarim & Kingsman's model underestimates cost — T&K $E\{TC\}$ is on average 5.26% lower than T&K $\widehat{E}\{TC\}$ — over the whole test bed the average difference between T&K $\widehat{E}\{TC\}$ and Exact $E\{TC\}$ is only 1.25%. This means that the approximate approach in [89] actually computes near-optimal parameters for (R^n,S^n) policy, reorder points and the respective order-up-to-levels, regardless of the underestimated cost. Nevertheless for some instances,

i.e. (29), (30), (31) etc., T&K $E\{TC\}$ is equal to T&K $\widehat{E}\{TC\}$, which means that for these instances the assumptions adopted by Tarim and Kingsman are valid. In summary these results suggest that Tarim & Kingsman's model can actually compute near-optimal policy parameters, although the approximate expected total cost predicted can often differ significantly from the actual expected total cost associated with these reorder points and respective order-up-to-levels.

As we may notice from the run-times reported in columns "sec", the approach proposed in [87] always outperforms our exact method and runs efficiently for every instance considered. Further results presented in [87] suggest that such an approach can efficiently handle large scale instances. Since our results suggest that the exact solution in the average case differs only slightly from the one provided by Tarim and Kingsman's approximate approach, when efficiency is an issue, their approach remains a valid alternative to our exact model.

2.6 Conclusions

We identified two sources of approximation in Tarim & Kingsman's model for computing (R^n,S^n) policy parameters under service level constraint. We proposed an exact *stochastic constraint programming* approach based on a novel concept — *global chance-constraints* — which extends the original stochastic constraint programming framework proposed by Walsh. We described a dedicated global chance-constraint that computes optimal inventory levels to meet the required service level and the expect total cost associated with them. We analyzed the accuracy of the approximate solutions provided by the model developed by Tarim & Kingsman over four different demand patterns and over several different input parameter configurations. We also provided insights into for which kind of instances the assumptions adopted by Tarim & Kingsman may affect the quality of the solution provided by their model. Our results suggest that their modeling strategy is a good trade-off between quality of the solution and efficiency of the search process.

Chapter 3

Paper II: Computing Replenishment Cycle Policy under Non-stationary Stochastic Lead Time

R. Rossi, S. A. Tarim, B. Hnich and S. Prestwich

Abstract

In this paper we address the general multi-period production/inventory problem with non-stationary stochastic demand and supplier lead time under service-level constraints. A replenishment cycle policy (R^n,S^n) is modeled, where R^n is the n-th replenishment cycle length and S^n is the respective order-up-to-level. Initially, we extend an existing formulation for this policy in such a way to incorporate a dynamic deterministic lead time allowing order-crossovers. Following this, we extend the model to incorporate a non-stationary stochastic lead time. Within a constraint programming framework, a dedicated constraint implementing a hybrid approach is proposed to compute replenishment cycle policy parameters.

3.1 Introduction

Inventory theory provides methods for managing inventories in different environments. An interesting class of production/inventory control problems is the one that considers the single location, single product case under non-stationary stochastic demand. In contrast to the production planning problem under deterministic demand (Wagner and Whitin [96]), different inventory control policies can be adopted to cope with the stochastic version.

A policy states the rules to decide when orders have to be placed and how to compute the replenishment lot-size for each order. For a discussion on inventory control policies see Silver et al. [81]. One of the well-known policies that can be adopted in inventory control is the replenishment cycle policy, (R,S). Under the non-stationary demand assumption this policy takes the dynamic form (R^n,S^n) where R^n denotes the length of the nth replenishment cycle, and S^n the order-up-to-level value for the nth replenishment.

It is a known result (Scarf [76]) that such a policy is not optimal in term of cost minimization, since non-stationary (s^n,S^n) always dominates it even when a delivery lag is considered (Kaplan [55]). However, as discussed in Tarim and Kingsman [89], (R, S) provides an effective means of dampening the planning instability. Furthermore, it is particularly appealing when items are ordered from the same supplier or require resource sharing. In such a case all items in a coordinated group can be given the same replenishment period. Periodic review also allows a reasonable prediction of the level of the workload on the staff involved and is particularly suitable for advanced planning environments. For these reasons, as stated by Silver et al. [81], (R, S) is a popular inventory policy.

Due to its combinatorial nature, (R^n,S^n) policy — even in the absence of stochastic lead time — presents a difficult problem to solve to optimality (Tarim and Kingsman [89]). Early work in the area have been carried out in Askin [3], Silver [80] and a heuristic procedure was proposed by Bookbinder and Tan [15]. Although many works in inventory control assume a penalty cost parameter for penalizing stock-outs, in all the works cited here the cost is minimized under a service level constraint, which is in practice a very popular measure, since it has been widely recognized that penalty costs, and in particular the cost of loosing

customer goodwill, are usually difficult to assess (Bashyam and Fu [6]).

A common assumption, in practice very restrictive, in all these works is the absence of delivery lag. A work on stochastic lead time in continuous-time inventory models was presented in Zipkin [101]. Kaplan [55] characterized the optimal policy for a dynamic inventory problem where the time lag in delivery of an item is a discrete random variable with known distribution. Since tracking all the outstanding orders by means of dynamic programming requires a large multi-dimensional state vector, Kaplan assumes that orders do not cross in time and that supplier lead time probabilities are independent of the size/number of outstanding orders (for details on order-crossover see Hayya et al. [43]). Under these assumptions he was able to provide a solution method for the problem and to derive the optimal policy. The first assumption is valid for systems where supplier's production system has a single-server queue structure operating under a FIFO policy. In Bashyam and Fu [6] a similar problem — operating under (s, S) policy, having a service level constraint and allowing orders to cross in time — is described and solved by means of a simulation based approach. To the best of our knowledge, there is no complete approach in the literature that addresses the (R^n,S^n) policy under stochastic supplier lead time.

In this paper, we use a "stochastic constraint programming" approach to address (R^n,S^n) policy under stochastic supplier lead time. Computing optimal policy parameters under these assumptions is a hard problem from a computational point of view. We build on the work of Eppen and Martin [27] and following a similar approach we develop a *scenario based method* [11, 91] for solving (R^n,S^n) under stochastic demand and supplier lead time. Efficient methods for computing (R^n,S^n) policy parameters based on Constraint Programming were proposed in Tarim et al. [87, 92]. In this paper, under the same assumptions, we develop a dedicated *constraint* that realizes a deterministic equivalent modeling of chance-constraints [18] by employing a scenario based approach [91]. A *constraint programming* (CP) [1] model is proposed and an example is given where an inventory control problem is solved to optimality under a given discrete stochastic supplier lead time with known distribution.

The paper is organized as follows. In Section 3.2 we provide some formal background related to the modeling techniques employed. In Section 3.3 we pro-

vide a formal definition for the general multi-period production/inventory problem with non-stationary stochastic demand and lead time. In Section 3.4 we extend Tarim and Kingsman's [89] model for the replenishment cycle policy in order to consider a dynamic deterministic supplier lead time, which assumes that orders may cross in time. In Section 3.5 former results are embedded in a scenario based approach to solve the problem when a stochastic supplier lead time with known probability mass function is given. In Section 3.6 a CP model is proposed, which incorporates former results in a dedicated constraint able to dynamically enforce the given service level constraint during search. Furthermore a demonstrative example is given in this section to clarify the approach. In Section 3.7 an instance is solved under deterministic and stochastic supplier lead times; solutions are then discussed. In Section 3.8 results are summarized and directions for future research are given.

3.2 Constraint Programming

A *Constraint Satisfaction Problem* (CSP) [1, 17, 62] is a triple $\langle V, C, D \rangle$, where V is a set of decision variables, D is a function mapping each element of V to a domain of potential values, and C is a set of constraints stating allowed combinations of values for subsets of variables in V. A *solution* to a CSP is simply a set of values of the variables such that the values are in the domains of the variables and all of the constraints are satisfied. We may also be interested in finding a feasible solution that minimizes (maximizes) the value of a given objective function over a subset of the variables. Alternatively, we can define a constraint as a mathematical function: $f : D_1 \times D_2 \times \ldots \times D_n \rightarrow \{0, 1\}$ such that $f(x_1, x_2, \ldots, x_n) = 1$ if and only if $C(x_1, x_2, \ldots, x_n)$ is satisfied. Using this functional notation, we can then define a constraint satisfaction problem (CSP) as follows (see also [1]): given n domains D_1, D_2, ..., D_n and m constraints f_1, f_2, ..., f_m find x_1, x_2, ..., x_n such that

$$f_k(x_1, x_2, \ldots, x_n) = 1, \quad 1 \leq k \leq m; \tag{3.1}$$

$$x_j \in D_j, \qquad 1 \leq j \leq n. \tag{3.2}$$

The problem is only a feasibility problem, and no objective function is defined. Nevertheless, CSPs are also an important class of combinatorial optimization problems. Here the functions f_k do not necessarily have closed mathematical forms (for example, functional representations) and can be defined simply by providing the set S described above.

For key concepts in Constraint Programming (CP) such as constraint filtering algorithm, constraint propagation and arc-consistency see [1, 67].

In [98] and [91] a *stochastic constraint satisfaction problem* (stochastic CSP) is defined as a 6-tuple $< V, S, D, P, C, \theta >$. V is a set of decision variables and S is a set of stochastic variables. D is a function mapping each element of V and each element of S to a domain of potential values. A decision variable in V is *assigned* a value from its domain. P is a function mapping each element of S to a probability distribution for its associated domain. C is a set of constraints. A constraint $h \in C$ that constrains at least one variable in S is a *chance-constraint*. θ_h is a threshold value in the interval $[0, 1]$, indicating the minimum satisfaction probability for chance-constraint h. Note that a chance-constraint with a threshold of 1 is equivalent to a hard constraint.

In [98] a policy based view of stochastic constraint programs is proposed. The semantics is based on a tree of decisions. Each path in a policy represents a different possible scenario (set of values for the stochastic variables), and the values assigned to decision variables in this scenario. To find satisfying policies, backtracking and forward checking algorithms, which explores the implicit AND/OR graph, are presented. Such an approach has been further investigated in [5]. An alternative semantics for stochastic constraint programs, which suggests an alternative solution method, comes from a scenario-based view [11]. In [91] the authors outline this solution method, which consists in generating a scenario-tree that incorporates all possible realizations of discrete random variables into the model explicitly. The great advantage of such an approach is that conventional constraint solvers can be used to solve stochastic CSP. Of course, there is a price to pay in this approach, as the number of scenarios grows exponentially with the number of stages and such a growth is particularly affected by random variables that contain a wide range of values in their domain.

3.3 Problem Definition

We consider a finite planning horizon of N periods and a demand d_t for each period $t \in \{1, \ldots, N\}$, which is a random variable with probability density function $g_t(d_t)$. We assume that the demand occurs instantaneously at the beginning of each time period. The demand we consider is non-stationary, that is it can vary from period to period, and we also assume that demands in different periods are independent.

In the following sections we will consider two different cases, respectively: a deterministic lead time of length L_t for an order placed in period $t \in \{1, ..., N\}$ and a stochastic lead time l_t with probability mass function $f_t(l_t)$ for an order placed in period $t \in \{1, ..., N\}$. Note that $\{l_t\}$ are mutually independent and each of them is also independent of the respective order quantity. A fixed delivery cost a is incurred for each order and a variable unit cost v. A linear holding cost h is incurred for each unit of product carried in stock from one period to the next. We assume that it is not possible to sell back excess items to the vendor at the end of a period and that negative orders are not allowed, so that if the actual stock exceeds the order-up-to-level for that review, this excess stock is carried forward and not returned to the supply source. However, such occurrences are regarded as rare events and accordingly the cost of carrying excess stocks and the positive effect on the service level of subsequent periods is ignored. As a service level constraint we require the probability that at the end of each and every period the net inventory will not be negative set to be at least a given value α. Our aim is to minimize the expected total cost, which is composed of ordering costs, unit costs and holding costs, over the N-period planning horizon, satisfying the service level constraints.

The actual sequence of ordering and delivery to be considered can be arbitrary as Kaplan notices in [55]. In the following we will adopt the same sequence of action he describes, since it handles all the deliveries symmetrically and allows for some delay in the arrival deliveries at the beginning of a period. The sequence is therefore as follows. At the beginning of a period, the inventory on hand after all the demands from previous periods have been realized is known. Since we are assuming complete backlogging, this quantity may be negative. Also known are

orders placed in previous periods which have not been delivered yet. On the basis of this information, an ordering decision is made for the current period. All the deliveries that are to be made during a period are assumed to be made immediately after this ordering decision and hence are on hand at the beginning of the period. A further discussion that states the convenience of this sequence of events can be found in Kaplan [55]. To summarize there are three successive events at the beginning of each period. First, stock on hand and outstanding orders are determined. Second, an ordering decision is made on the basis of this information. Third, all supplier deliveries for the current period, including possibly the most recent orders, are received.

3.4 Dynamic Deterministic Lead Time

In this section we focus on the general multi-period production/inventory problem with stochastic demands and dynamic deterministic lead time. The reader may also refer to [42] about this topic. This problem can be formulated as finding the timing of the stock reviews and the size of the respective non-negative replenishment orders, X_t in period t, with the objective of minimizing the expected total cost $E\{TC\}$ over a finite planning horizon of N periods. Since a dynamic deterministic lead time $L_t \geq 0$ is considered in each period $t = 1, \ldots, N$, an order placed in period t will be received only at period $t + L_t$. Depending on the values assigned to L_t it may be obviously not possible to provide the required service level for some initial periods. In general we will be able to provide the required service level α starting from the period t for which the value $t + L_t$ is minimum. Let M be this period. Notice also that it will never be optimal to place any order in a period t such that $t + L_t > N$, since such an order will not be received within the given planning horizon. The problem can be formulated as a chance-constrained programming model (see Bookbinder and Tan [15]),

$$\begin{aligned} \min \quad E\{TC\} = \int_{d_1}\int_{d_2}\ldots\int_{d_N} \sum_{t=1}^{N} (a\delta_t + vX_t + h \cdot \max(I_t, 0)) \\ \times g_1(d_1)g_2(d_2)\ldots g_N(d_N)\mathrm{d}(d_1)\mathrm{d}(d_2)\ldots\mathrm{d}(d_N) \end{aligned} \tag{3.3}$$

subject to,

$$\delta_t = \begin{cases} 1, & \text{if} \quad X_t > 0 \\ 0, & \text{otherwise} \end{cases} \qquad t = 1, ..., N \qquad (3.4)$$

$$I_t = I_0 + \sum_{\{i|i\geq 1, L_i+i\leq t\}} X_i - \sum_{i=1}^{t} d_i \qquad t = 1, ..., N \qquad (3.5)$$

$$\Pr\{I_t \geq 0\} \geq \alpha \qquad t = M, ..., N \qquad (3.6)$$

$$I_t \in \mathbb{Z}, \quad X_t \geq 0, \quad \delta_t \in \{0, 1\} \qquad t = 1, ..., N \qquad (3.7)$$

where we comply with the notation used in [15],

d_t : the demand in period t, a random variable with probability density function, $g_t(d_t)$,

a : the fixed ordering cost (incurred when an order is placed),

h : the proportional stock holding cost,

v : the unit variable cost of an item,

L_t : the deterministic delivery lead time in period t, $L_t \geq 0$

δ_t : a $\{0,1\}$ variable that takes the value of 1 if a replenishment occurs in period t and 0 otherwise,

I_t : the inventory level (stock on hand minus back-orders) at the end of period t,

I_0 : the initial inventory,

X_t : the size of the replenishment order placed in period t, $X_t \geq 0$, (received in period $t + L$).

Let us denote the inventory position (the total amount of stock on hand plus outstanding orders minus back-orders) at the end of period t as P_t. It directly follows that

$$P_t = I_t + \sum_{\{i|1\leq i\leq t, L_i+i>t\}} X_i. \qquad (3.8)$$

where P_t is the inventory position in period t and it is assumed $P_0 = I_0$. We now reformulate the model using the inventory position,

$$\min \quad E\{TC\} = \int_{d_1}\int_{d_2}\cdots\int_{d_N}\sum_{t=1}^{N}\left(a\delta_t + vX_t + h\cdot\max(P_t - \sum_{\{i|1\le i\le t, L_i+i>t\}} X_i, 0)\right) \times g_1(d_1)g_2(d_2)\ldots g_N(d_N)\mathrm{d}(d_1)\mathrm{d}(d_2)\ldots\mathrm{d}(d_N) \tag{3.9}$$

subject to,

$$\delta_t = \begin{cases} 1, & \text{if} \quad X_t > 0 \\ 0, & \text{otherwise} \end{cases} \qquad t = 1, ..., N \tag{3.10}$$

$$P_t = I_0 + \sum_{i=1}^{t}(X_i - d_i) \qquad t = 1, ..., N \tag{3.11}$$

$$\Pr\{P_t \ge \sum_{\{i|1\le i\le t, L_i+i>t\}} X_t\} \ge \alpha \qquad t = M, ..., N \tag{3.12}$$

$$P_t \in \mathbb{Z}, \quad X_t \ge 0, \quad \delta_t \in \{0, 1\} \qquad t = 1, ..., N. \tag{3.13}$$

By using the expectation operator $E\{\cdot\}$, since $\{d_t\}$ are assumed to be mutually independent, we may rewrite the objective function as

$$\min \quad E\{TC\} = \sum_{t=1}^{N}\left(h\cdot E\left\{\max(P_t - \sum_{\{i|1\le i\le t, L_i+i>t\}} X_i, 0)\right\} + a\cdot\delta_t + v\cdot X_t\right). \tag{3.14}$$

When a stock-out occurs, all demand is back-ordered and filled as soon as an adequate supply arrives. However, the probability that net inventory will not be negative is set normally quite high by the management, so that the cost of back-orders can be ignored in the model. Moreover, Bookbinder and Tan discuss that the term $E\{\max(I_t, 0)\}$ may be approximated by $E\{I_t\}$, in view of these remarks. Therefore in our model we approximate the term $E\{\max(P_t - \sum_{\{i|1\le i\le t, L_i+i>t\}} X_i, 0)\}$ with the term $E\{P_t - \sum_{\{i|1\le i\le t, L_i+i>t\}} X_i\}$.

The general chance constrained programming formulation given above can be modified to incorporate the inventory control policy adopted. In this paper we adopt the "replenishment cycle policy", which is equivalent to Bookbinder-Tan's "static-dynamic uncertainty strategy". The replenishment cycle policy (ie, (R, S) policy) is *static* in the sense that the replenishment periods are determined once and for all at the beginning of the planning horizon, and *dynamic* as the order quantities are decided only after observing the realized demand. In what follows –based on [89], in which lead times are ignored– we formulate the replenishment cycle policy under dynamic deterministic lead times, L_t.

Consider a review schedule, which has m reviews over the N period planning horizon with orders placed at $\{T_1, T_2, \ldots, T_m\}$, where $T_i > T_{i-1}$, $T_m \leq N - L_{T_m}$. For convenience T_1 is defined as the start of the planning horizon and $T_{m+1} = N + 1$ as the period immediately after the end of the planning horizon. The review schedule may be generalized to consider the case where $T_1 > 1$, if the opening stock I_0 is sufficient to cover the immediate needs at the start of the planning horizon. The associated stock reviews will take place at the beginning of periods T_i, $i = 1, \ldots, m$. In the considered dynamic review and replenishment policy clearly the orders X_i are all equal to zero except at replenishment periods $T_1, T_2, \ldots, T_m$. The inventory level I_t carried from period t to period $t + 1$ is the opening stock plus any orders that have arrived up to and including period t less the total demand to date. Hence is given by

$$I_t = I_0 + \sum_{\{i|L_{T_i}+T_i \leq t\}} X_{T_i} - \sum_{k=1}^{t} d_k, \quad t = 1, \ldots, N. \tag{3.15}$$

Let us define

$$p(t) = \max \left\{ i | \forall j, j \leq i, T_j + L_{T_j} \leq t, \quad i = 1, \ldots, m \right\}. \tag{3.16}$$

The inventory level I_t at the end of period t (Eq. 3.15) can be expressed as

$$I_t = I_0 + \sum_{i=1}^{p(t)} X_{T_i} + \sum_{\{i|i>p(t), L_{T_i}+T_i \leq t\}} X_{T_i} - \sum_{k=1}^{t} d_k, \quad t = 1, \ldots, N. \tag{3.17}$$

We now want to reformulate the constraints of the chance constrained model in terms of a new set of decision variables R_{T_i}, $i = 1, \ldots, m$. We define

$$P_t = R_{T_i} - \sum_{k=T_i}^{t} d_k, \quad T_i \leq t < T_{i+1}, \quad i = 1, \ldots, m \tag{3.18}$$

where R_{T_i} can be interpreted as an order-up-to-position which stock should be raised after placing an order at the ith review period T_i, and $R_{T_i} - \sum_{k=T_i}^{t} d_k$ is the end of period inventory position. We can now express the whole model in term of these new decision variables R_{T_i}, which are related to the inventory position in period T_i. The new problem is therefore to determine the number of reviews, m, the T_i, and the associated R_{T_i} for $i = 1, \ldots, m$.

If there is no replenishment scheduled for period t, then R_t equals the opening inventory position in period t. It follows that the variable R_t must be equal to P_{t-1} if no order is placed in period t and equal to the order-up-to-position if there is a review in period t. We can express this using the following constraints

$$R_t = P_t + d_t, \qquad t = 1, \ldots, N \tag{3.19}$$

$$R_t \geq P_{t-1}, \qquad t = 1, \ldots, N \tag{3.20}$$

$$R_t > P_{t-1} \Rightarrow \delta_t = 1, \quad t = 1, \ldots, N. \tag{3.21}$$

The values for the order-up-to-position variables, R_t, are then those that give the minimum expected total cost $E\{TC\}$. The desired opening stock positions, as required for the solution to the problem, will then be those values of R_t, for which $\delta_t = 1$. It is now clear that Constraints 3.4 and 3.5 can be replaced by Eq. 3.19, 3.21 and 3.20.

Let us now express Eq. 3.17 using R_{T_i} as decision variables

$$I_t = R_{T_{p(t)}} + \sum_{\{i | i > p(t), L_{T_i} + T_i \leq t\}} \left(R_{T_i} - R_{T_{i-1}} + d_{T_{i-1}} + \ldots + d_{T_i - 1}\right) - \sum_{k=T_{p(t)}}^{t} d_k, \quad t = 1, \ldots, N. \tag{3.22}$$

As already mentioned, α is the desired minimum probability that the net inventory level in any time period will be non-negative. M is by definition the first period at which the inventory can be controlled. Keeping this in mind we require

$$\Pr\{I_t \geq 0\} \geq \alpha, \quad t = M, \ldots, N. \tag{3.23}$$

which implies, by substituting I_t with the right term in Eq. 3.22,

$$G_S\left(R_{T_{p(t)}} + \sum_{\{i|i>p(t), L_{T_i}+T_i \leq t\}} (R_{T_i} - R_{T_{i-1}})\right) \geq \alpha, \quad t = M, \ldots, N. \tag{3.24}$$

where $S = \sum_{k=T_{p(t)}}^{t} d_k - \sum_{\{i|i>p(t), L_{T_i}+T_i \leq t\}} (d_{T_{i-1}} + \ldots + d_{T_i-1})$ and, as given in [15], $G_{d_1+d_2+\ldots+d_t}(.)$ is the cumulative distribution function of $D(t) = d_1 + d_2 + \ldots + d_t$.

We now express the whole model in terms of the new set of variables R_i. Since we consider expectations $\tilde{P}_i$ and $\tilde{d}_i$, it follows that $R_i = \tilde{P}_i + \tilde{d}_i$ and also that the term X_t in the objective function can be expressed as $R_t - \tilde{P}_{t-1}$. We replace the service level constraint 3.6 using the new formulation in Eq. 3.24. We should note that $v \sum_{t=1}^{N} \left(R_t - \tilde{P}_{t-1}\right)$ in the objective function can be rewritten as $v \sum_{t=1}^{N} \tilde{d}_t + v \cdot P_N$, where $\sum_{t=1}^{N} \tilde{d}_t$ is obviously a constant of the problem. The resulting model is as follows,

$$E\{TC\} =$$

$$v \sum_{t=1}^{N} \tilde{d}_t + \min \left[\sum_{t=1}^{N} \left(h \cdot \left(\tilde{P}_t - \sum_{\{i|1\leq i\leq t, L_i+i>t\}} (R_i - \tilde{P}_{i-1})\right) + a \cdot \delta_t\right) + v \cdot \tilde{P}_N\right] \tag{3.25}$$

subject to,

$$\{T_1, \ldots, T_m\} = \{t \in \{1, \ldots, N\} | \delta_t = 1\}$$
$$\text{Eq. } 3.24, \qquad t = M, \ldots, N$$

$$R_t > \tilde{P}_{t-1} \Rightarrow \delta_t = 1, \qquad t = 1, \ldots, N \tag{3.26}$$
$$R_t \geq \tilde{P}_{t-1}, \qquad t = 1, \ldots, N \tag{3.27}$$
$$R_t = \tilde{P}_t + \tilde{d}_t, \qquad t = 1, \ldots, N \tag{3.28}$$
$$R_t \geq 0, \quad \tilde{P}_t \geq 0, \quad \delta_t \in \{0, 1\} \quad t = 1, \ldots, N \tag{3.29}$$

So far we treated the replenishment cycle policy formulation of the production/inventory problem under non-stationary stochastic demand, d_t, and dynamic deterministic lead time, L_t. We now recall that a deterministic equivalent formulation of this problem under the same policy, non-stationary stochastic demand, d_t, and deterministic but constant lead time, L, was proposed in [86]. According to this formulation and from the results presented here, when the lead time is deterministic and constant, it is easy to see that Eq. 3.24 becomes

$$G_{d_{T_{p(t)}} + d_{T_{p(t)}+1} + \ldots + d_t}(R_{T_{p(t)}}) \geq \alpha, \quad t = L+1, \ldots, N. \tag{3.30}$$

We adopt the following change of variable: $T_i = T_{p(t)}$. Since the lead time is deterministic and constant T_i will be equal to $T_{p(t)}$ for every t such that $T_i + L \leq t < T_{i+1} + L$. It directly follows that

$$G_{d_{T_i} + d_{T_i+1} + \ldots + d_t}(R_{T_i}) \geq \alpha, \quad T_i + L \leq t < T_{i+1} + L. \tag{3.31}$$

By defining $k = t - L$ we can rewrite the former expression as

$$G_{d_{T_i} + d_{T_i+1} + \ldots + d_{k+L}}(R_{T_i}) \geq \alpha, \quad T_i \leq k < T_{i+1} \tag{3.32}$$

and therefore, since $\tilde{P}_k = R_{T_i} - \sum_{n=T_i}^{k} \tilde{d}_n$, it follows,

$$\tilde{P}_k \geq G^{-1}_{d_{T_i}+d_{T_i+1}+\ldots+d_{k+L}}(\alpha) - \sum_{n=T_i}^{k} \tilde{d}_n, \quad T_i \leq k < T_{i+1}. \tag{3.33}$$

G^{-1} is an "inverse function", such that $G^{-1}_{D(t)}(\alpha) = u$ means $\alpha = G_{D(t)}(u) = \Pr\{D(t) \leq u\}$. We assume that G is strictly increasing, hence G^{-1} is uniquely defined. The right-hand side of Eq. 3.33 can be calculated off-line and memorized in a table once the form of $g_t(\cdot)$ is selected. Let

$$\Phi[i,j] = G^{-1}_{d_i+d_{i+1}+\ldots+d_{j+L}}(\alpha) - \sum_{k=i}^{j} \tilde{d}_k. \tag{3.34}$$

By employing the table presented in Eq. 3.34, the whole model under deterministic and constant lead time, L, can be easily expressed using a CP formulation similar to the one presented in [92]. The whole model is

$$E\{TC\} =$$

$$v \sum_{t=1}^{N} \tilde{d}_t + \min \quad \left[\sum_{t=1}^{N} \left(h \cdot \left(\tilde{P}_t - \sum_{i=t-L+1}^{t} (R_i - \tilde{P}_{i-1}) \right) + a \cdot \delta_t \right) + v \cdot \tilde{P}_N \right] \tag{3.35}$$

subject to,

$$R_t > \tilde{P}_{t-1} \Rightarrow \delta_t = 1 \qquad t = 1, \ldots, N \tag{3.36}$$

$$R_t \geq \tilde{P}_{t-1} \qquad t = 1, \ldots, N \tag{3.37}$$

$$\tilde{P}_t \geq \Phi[\max_{j \in \{1..t\}} \{j \cdot \delta_j\}, t] \qquad t = 1, \ldots, N-L \tag{3.38}$$

$$R_t = \tilde{P}_t + \tilde{d}_t, \qquad t = 1, \ldots, N \tag{3.39}$$

$$R_t \geq 0, \quad \tilde{P}_t \geq 0, \quad \delta_t \in \{0,1\} \qquad t = 1, \ldots, N \tag{3.40}$$

where elements in matrix Φ are indexed using the `element` constraint [45]. Obviously if we want to invert the cumulative distribution function in Eq. 3.24 as in the constant lead time case, the dimension of the table where the buffer stock

levels are stored has to increase, since many decision variables take part in the computation of the stock-out probability. Instead of building this matrix, it may be therefore convenient to develop a dedicated *constraint* for the CP formulation of the model. In fact, in CP relations between decision variables can be expressed by means of dedicated constraints that may include customized algorithms to generate parameters and verify complex conditions like Eq. 3.24. In this constraint we simply wait for a partial assignment of decision variables $\{\delta_t\}$ and, by using Eq. 3.24, we dynamically generate during the search deterministic equivalent constraints in a way similar to the one presented in the example above. These deterministic constraints are enforced to guarantee the required service level under the given partial replenishment plan.

3.5 Non-stationary Stochastic Lead Time

We now consider the general multi-period production/inventory problem with non-stationary stochastic demand and lead time. As in Eppen and Martin [27], we consider a discrete stochastic lead time with probability mass function $f_i(\cdot)$ in each period $i = 1, \ldots, N$. This means that an order placed in period i will be received after k periods with probability $f_i(k)$. Since $f_i(k)$ is discrete we shall assume that there is a maximum lead time L for which $\sum_{k=0}^{L} f_i(k) = 1$, $i = 1, \ldots, N$. The probability of observing any lead time length $p > L$ will be always 0. Therefore the possible lead time lengths are limited to $S = \{0, \ldots, L\}$ and the probability mass function is defined on the finite set S. Depending on the probabilities assigned to each lead time length by the probability mass function, it may not be possible to provide the required service level for some initial periods. In general, reasoning in a worst case scenario, it will always be possible to provide the required service level α starting from period $L + 1$. The chance-constrained programming model is given below,

$$\begin{aligned} \min \quad E\{TC\} = \int_{d_1} \ldots \int_{d_N} \sum_{l_1} \ldots \sum_{l_N} \sum_{t=1}^{T} (v \cdot X_t + a \cdot \delta_t + h \cdot I_t) \\ f_1(l_1) f_2(l_2) \ldots f_N(l_N) \times g_1(d_1) g_2(d_2) \ldots g_N(d_N) d(d_1) d(d_2) \ldots d(d_N) \end{aligned} \tag{3.41}$$

subject to,

$$I_t = I_0 + \sum_{\{i|i\geq 1, l_i \leq t-i\}} X_i - d_t \qquad t = 1, \ldots, N \qquad (3.42)$$

$$\delta_t = \begin{cases} 1, & \text{if} \quad X_t > 0 \\ 0, & \text{otherwise} \end{cases} \qquad t = 1, \ldots, N \qquad (3.43)$$

$$\Pr\{I_t \geq 0\} \geq \alpha \qquad t = L+1, \ldots, N \qquad (3.44)$$

$$I_t \in \mathbb{Z}_0^+, \quad X_t \geq 0, \quad \delta_t \in \{0, 1\} \qquad t = 1, \ldots, N \qquad (3.45)$$

where

l_i : the lead time length of the order placed in period i, a discrete random variable with probability mass function $f_i(\cdot)$.

We now reformulate the model using the inventory position,

$$\min \quad E\{TC\} =$$

$$\int_{d_1} \ldots \int_{d_N} \sum_{l_1} \ldots \sum_{l_N} \sum_{t=1}^{N} \left(a\delta_t + vX_t + h \cdot \left(P_t - \sum_{\{i|1\leq i\leq t, l_i > t-i\}} X_i \right) \right)$$
$$f_1(l_1) f_2(l_2) \ldots f_N(l_N) \times g_1(d_1) g_2(d_2) \ldots g_N(d_N) \mathrm{d}(d_1) \mathrm{d}(d_2) \ldots \mathrm{d}(d_N) \qquad (3.46)$$

subject to,

$$\delta_t = \begin{cases} 1, & \text{if} \quad X_t > 0 \\ 0, & \text{otherwise} \end{cases} \qquad t = 1, ..., N \qquad (3.47)$$

$$P_t = I_0 + \sum_{i=1}^{t} (X_i - d_i) \qquad t = 1, ..., N \qquad (3.48)$$

$$\Pr\{P_t \geq \sum_{\{i|1\leq i\leq t, l_i > t-i\}} X_i\} \geq \alpha \qquad t = L+1, ..., N \qquad (3.49)$$

$$P_t \in \mathbb{Z}_0^+, \quad X_t \geq 0, \quad \delta_t \in \{0, 1\} \qquad t = 1, ..., N. \qquad (3.50)$$

Let us define the cumulative distribution function $F_i(k) = \sum_{p=0}^{k} f_i(p)$, $k \geq 0$. Given the probability mass function $f_i(l_i)$ and since l_i is a discrete random variable it directly follows

$$\sum_{i=1}^{t} F_i(t-i)X_i = \sum_{i=1}^{t} \sum_{p=0}^{t-i} f_i(p)X_i \qquad t = 1, \ldots, N. \tag{3.51}$$

By recalling that $\{d_t\}$ are assumed to be mutually independent, we may rewrite the objective function as

$$\min \quad E\{TC\} =$$

$$\sum_{t=1}^{N} \left(h \cdot E \left\{ \left(P_t - \sum_{i=1}^{t} (1 - F_i(t-i))X_i \right) + v \cdot X_t \right\} + a \cdot \delta_t \right) \tag{3.52}$$

Also in this case we want to adopt a replenishment cycle policy and we want to express the whole model in terms of the new set of variables R_i, so that order quantities have to be decided only after the demand in the former periods have been realized. The analysis developed in the former section for the replenishment condition (Eq. 3.43) and inventory conservation constraints (Eq. 3.42) still holds, since it refers to the opening-inventory-position, which by definition is not affected by the lead time length. So it is clear that these constraints can be replaced by Eq. 3.19, 3.21 and 3.20. Since we are considering expectations, the term X_t in the objective function can be expressed as $R_t - \tilde{P}_{t-1}$. As we did in the dynamic deterministic lead time case, we now have to express the service level constraint as a relation between the opening-inventory-positions such that the overall service level provided at the end of each period is at least α. In order to express this service level constraint we propose a scenario based approach over the discrete random variables l_i, $i = 1, \ldots, N$. Let us recall that in a scenario based approach [11, 91], a scenario tree is generated which incorporates all possible realization of discrete random variables into the model explicitly. A path from the root to an extremity of the event tree represents a scenario $\omega \in \Omega$, where Ω is the set of all possible scenarios. To each scenario a given probability is associated. If S_i is the ith random variable on a path from the root to the leaf representing scenario ω and

a_i is the value given to S_i in the ith stage of this scenario, then the probability of this scenario is given by $\Pr\{\omega\} = \prod_i \Pr(S_i = a_i)$. Within each scenario, we have a conventional (non-stochastic) constraint program to solve. All we have to do is replacing the stochastic variables by the values taken in the scenario and ensure that the values found for the decision variables are consistent across scenarios as certain decision variables are shared across scenarios.

In our problem we can divide random variables into two sets: the discrete random variables $\{l_i\}$ which represent lead times and the continuous random variables $\{d_i\}$ which represent demands. We deal with each set in a separate fashion, by employing a scenario based approach for the discrete random variables and a deterministic equivalent modeling approach for the continuous random variables. This is possible since, as we have already remarked, under a given scenario ω discrete random variables are treated as deterministic values. The problem is then reduced to the general multi-period production/inventory problem with dynamic deterministic lead time and stochastic demand, for which we have already presented in the former section a deterministic equivalent model that is able to represent the chance-constraints involving continuous random variables $\{d_i\}$.

Consider a review schedule Z, which has m reviews over the N period planning horizon with orders placed at $\{T_1, T_2, \ldots, T_m\}$, where $T_i > T_{i-1}$, $T_m \leq N$. For convenience T_1 is defined as the start of the planning horizon and $T_{m+1} = N + 1$ as the period immediately after the end of the planning horizon. The review schedule may be generalized to consider the case where $T_1 > 1$, if the opening stock I_0 is sufficient to cover the immediate needs at the start of the planning horizon. The associated stock reviews will take place at the beginning of periods T_i, $i = 1, \ldots, m$. In the considered dynamic review and replenishment policy clearly the orders X_i are all equal to zero except at replenishment periods $T_1, T_2, \ldots, T_m$. The inventory level I_t carried from period t to period $t + 1$ is the opening stock plus any orders that have arrived up to and including period t less the total demand to date. A scenario ω_t is a possible lead time realization for all the orders placed up to period t in the given review schedule Z. Let Ω_t be the set of all the possible scenarios ω_t. The first observation we need is related to the definition of $p(t)$ (Eq. 3.16). We have defined $T_{p(t)}$ as the latest period before period t in the planning horizon, for which we are sure that all the former orders, including the one placed

in $T_{p(t)}$ if there is any, have been delivered within period t. Under the assumption that the probability mass function $f_i(\cdot)$ is defined on a finite set S, $p(t)$ provides a bound for the scenario tree size. In fact if the possible lead time lengths in S are $0, \ldots, L$, the earliest order that is delivered in period t with probability 1 under every possible scenario ω_t is the latest placed in the span $1, \ldots, t-L$. Therefore since each scenario ω_t identifies the orders that have been received before or in period t, it directly follows that the number of scenarios in the tree that is needed to compute the buffer stocks for periods $t-L, \ldots, t$ under any possible review schedule Z is at most 2^L, when we place $L+1$ orders in periods $t-L, \ldots, t$, but it may be lower if less reviews are planned. Under a given review schedule Z and a scenario ω_t the service level constraint for a period t can be easily expressed by means of Eq. 3.24. It follows that the service level constraint is always a relation between at most $L+1$ decision variables P_i that represent the closing-inventory-position (or equivalently R_i which are the order-up-to-position) of the replenishment cycles covering the span $t-L, \ldots, t$. Let $p_\omega(t)$ be the value of $p(t)$ under a given scenario ω_t when a review schedule Z is considered. In order to satisfy the service level constraints in our original model, we require that the overall service level under all the possible scenarios for each set of at most $L+1$ decision variables is at least α or equivalently, by using Eq. 3.24

$$\sum_{\omega_t \in \Omega_t} \Pr\{\omega_t\} \cdot G_S \left(R_{T_{p_\omega(t)}} + \sum_{\{i | i > p_\omega(t), (l_{T_i}|\omega_t) \leq t - T_i\}} (R_{T_i} - R_{T_{i-1}}) \right) \geq \alpha, \quad t = L+1, \ldots, N, \tag{3.53}$$

where $S = \sum_{k=T_{p_\omega(t)}}^{t} d_k - \sum_{\{i | i > p_\omega(t), (l_{T_i}|\omega_t) \leq t - T_i\}} (d_{T_{i-1}} + \ldots + d_{T_i - 1})$. Therefore the complete model under the replenishment cycle policy can be expressed as

$$E\{TC\} =$$

$$v \sum_{t=1}^{N} \tilde{d}_t + \min \left[\sum_{t=1}^{N} \left(h \cdot \left(\tilde{P}_t - \sum_{i=1}^{t} (1 - F_i(t-i))(R_i - \tilde{P}_{i-1}) \right) + a \cdot \delta_t \right) + v \cdot \tilde{P}_N \right] \tag{3.54}$$

subject to,

$$\{T_1, \ldots, T_m\} = \{t \in \{1, \ldots, N\} | \delta_t = 1\}$$
$$\text{Eq. } 3.53, \qquad t = L+1, \ldots, N$$

$$R_t > \tilde{P}_{t-1} \Rightarrow \delta_t = 1 \qquad t = 1, \ldots, N \tag{3.55}$$
$$R_t \geq \tilde{P}_{t-1} \qquad t = 1, \ldots, N \tag{3.56}$$
$$R_t = \tilde{P}_t + \tilde{d}_t \qquad t = 1, \ldots, N \tag{3.57}$$
$$R_t \geq 0, \quad \tilde{P}_t \geq 0, \quad \delta_t \in \{0, 1\} \qquad t = 1, \ldots, N. \tag{3.58}$$

3.6 Stochastic Lead Time: a CP Implementation

In this section we present a CP formulation for the (R^n,S^n) problem under stochastic lead time. Results from the former section will be employed in the CP formulation. In order to model the service level constraint (Eq. 3.53) we presented in the former section, a new constraint *serviceLevel*($\cdot$) will be defined. Such a constraint is needed to dynamically compute the correct buffer stock positions on the basis of the current replenishment plan, that is $\{\delta_t\}$ assignments. Without loss of generality we will consider here a different and simpler objective function. In such a function we will charge a holding cost at the end of each period based on the current inventory position, rather than the current inventory level. This will reflect the fact that we charge interests not only on the actual amount of items we have in stock, but also on outstanding orders. It should be noted that it is possible to build a CP model that considers the original objective function. We chose not to implement this function in our tool. In fact, in the research project carried out for a leading international telecommunications company that motivated this research we were explicitly required to charge holding cost on the inventory position and not on the inventory level. Doing so often make sense since companies may assess holding cost on their total invested capital and not simply on items in stock. A further and detailed justification for this can be found in [48]†.

†In this work the author considers a holding cost based on the inventory position rather than on-hand inventory in their order-up-to policy. He underlines how a holding cost based on inventory

The CP model that incorporates our dedicated chance constraint and the objective function discussed is therefore

$$\min \quad E\{TC\} = \sum_{t=1}^{N} \left(a \cdot \delta_t + h \cdot \tilde{P}_t \right) + v \cdot \tilde{P}_N \tag{3.59}$$

subject to,

$$\tilde{P}_t + \tilde{d}_t - \tilde{P}_{t-1} > 0 \Rightarrow \delta_t = 1 \qquad t = 1, \ldots, N \tag{3.60}$$

$$\delta_t = 0 \Rightarrow \tilde{P}_t + \tilde{d}_t - \tilde{P}_{t-1} = 0 \qquad t = 1, \ldots, N \tag{3.61}$$

$$\tilde{P}_t + \tilde{d}_t - \tilde{P}_{t-1} \geq 0 \qquad t = 1, \ldots, N \tag{3.62}$$

$$\begin{aligned} serviceLevel(&\delta_1, \ldots, \delta_N, \\ &\tilde{P}_1, \ldots, \tilde{P}_N, \\ &g_1(d_1), \ldots, g_N(d_N), \\ &f(\cdot), \alpha) \end{aligned} \tag{3.63}$$

$$\tilde{P}_t \geq 0, \quad \delta_t \in \{0, 1\} \qquad t = 1, \ldots, N. \tag{3.64}$$

It must be noted that the domain size value for the $\tilde{P}_t$ variables, exactly as in the zero lead time case, is limited and more precisely it is equal to the amount of stock required to satisfy subsequent demands till the end of the planning horizon, meeting the required service level when only a single replenishment is scheduled at the beginning of the planning horizon. In what follows we describe the signature of the new constraint we have introduced. *serviceLevel*$(\cdot)$ describes a relation between all the decision variables in the model. It also accepts as parameters the

position provides a simple and more accurate expression for inventory holding costs in the combined manufacturing and warehouse divisions. In fact he observed that the order of a part initiates a succession of charges which are incurred throughout the lead time (direct material cost, direct labor cost and overheard cost). Certain inventory carrying costs are based on these charges – interest on investment and risk of obsolescence – and they are accrued from the time an order is placed to the manufacturing division. On the other hand other inventory carrying costs are accrued from the time the finished part is delivered to the warehouse (warehousing costs). The author suggests that a precise expression for the inventory carrying costs which reflected all these consideration would be very complex. Therefore, when interest and risk of obsolescence comprise a large portion of the total carrying cost, using a model which incurs carrying cost from the time an order is placed rather then from the time is delivered may be the correct choice

distribution of the demand in each period; the probability mass function of the lead time, which is assumed to be the same for all the periods; and the required service level. In order to enforce this constraint we consider every group of consecutive replenishment cycles that cover at least $L + 1$ periods (that is the one of interest plus L former periods). Each group must have the smallest possible cardinality in term of replenishment cycle number. Obviously, to identify this group of cycles, we have to wait that a subset of consecutive δ_t variables is assigned. Then, in order to verify if the service level constraint is satisfied for the last period in this group, we check that for each replenishment cycle in the group identified at least one decision variable $\tilde{P}_t$ is assigned. If this is the case the partial policy for the span is completely defined and, by recalling that $R_t = \tilde{P}_t + \tilde{d}_t$, its feasibility can be checked by using the condition in Eq. 3.53. If the condition is not satisfied we backtrack. Notice that such a condition involves only the periods we identified in the group defined, this means that our constraint is able to detect infeasibility of partial assignments. A high level pseudo-code for the propagation logic of the global chance-constraint described is presented in Algorithm 4. Note that to keep the description of the algorithm simple we assume here a stochastic lead time l with probability mass function $f(l)$ in every period. The maximum lead time length is L. It should be also emphasized that, during the search, any CP solver will be able to exploit constraint propagation and detect infeasible or suboptimal assignments with respect to other constraints in the model. Furthermore many infeasible or suboptimal solutions may be pruned by using respectively dedicated *forward checking* techniques like the one described in [98] or *cost-based filtering* methods [31, 87].

Example 3.6.1. We assume an initial null inventory level and a normally distributed demand with a coefficient of variation $\sigma_t/\tilde{d}_t = 0.3$ for each period $t \in \{1, \ldots, 5\}$. The expected values for the demand in each period are: $\{36, 28, 42, 33, 30\}$. The other parameters are $a = 1$, $h = 1$, $v = 0$, $\alpha = 0.95 (z_{\alpha=0.95} = 1.645)$. We consider for every period i in the planning horizon the following lead time probability mass function $f_i(t) = \{0.3, 0.2, 0.5\}$, which means that we receive an order placed in period i after $t \in \{0, \ldots, 2\}$ periods with the given probability (0 periods: 30%; 1 period: 20%; 2 periods: 50%). It is obvious that in this case we will always receive the order at most after 2 periods. In Table

Algorithm 4: `propagate`

input : $\delta_1, \ldots, \delta_N$, $\tilde{P}_1, \ldots, \tilde{P}_N$, α, $d_1, \ldots, d_N$, l, L, N

begin

- $cycles \leftarrow \{\}$;
- $pointer \leftarrow 1$;
- $periods \leftarrow 0$;
- **for each period** i **in** $2, \ldots, N$ **do**
  - **if** δ_i is not assigned **then**
    - $cycles \leftarrow \{\}$;
    - $periods \leftarrow 0$;
    - $pointer = -1$;
  - **else if** δ_i **is assigned to** 1 **then**
    - **if** $pointer \neq -1$ **then**
      - $cycle \leftarrow$ a replenishment cycle over $\{pointer, ..., i-1\}$;
      - add $cycle$ to $cycles$;
    - **if** $periods \geq L$ **then**
      - checkBuffers();
    - $pointer \leftarrow i$;
    - $periods \leftarrow periods + 1$;
  - **else**
    - $periods \leftarrow periods + 1$;
- **if** $pointer \neq -1$ **then**
  - $cycle \leftarrow$ a replenishment cycle over $\{pointer, ..., N\}$;
  - add $cycle$ to $cycles$;
- **if** $periods \geq L$ **then**
  - checkBuffers();

end

3.1 (Fig. 3.1) we show the optimal solution found when our chance constraint is used to dynamically generate buffer stock levels. We now want to show that order-up-to-positions computed in this example by using condition 3.53 satisfy every service level constraint in the model. We assume that for the first 2 periods no service level constraint is enforced, since it is not possible to fully control the inventory in the first 2 periods. Therefore we enforce the required service level on period 3, 4 and 5, that is constraint 3.53 for $t = 3, \ldots, N$. Let us verify that the given order-up-to levels satisfy this condition for each of these three periods.

Procedure `checkBuffers`

```
begin
    cycle ← the last element in cycles, a replenishment cycle over
    {i, ..., j};
    if no decision variable P̃_i, ..., P̃_j is assigned then
        return;
    counter ← 1;
    for each period t covered by cycle do
        formerCycles ← cycles;
        remove cycle from formerCycles;
        coveredPeriods ← the number of periods covered by cycles in
        formerCycles;
        head ← first element in formerCycles;
        headLength ← periods covered by head;
        if counter < L then
            while coveredPeriods − headLength + counter ≥ L do
                remove head from formerCycles;
                head ← first element in formerCycles;
                headLength ← periods covered by head;
        else
            formerCycles ← {};
        condition ← true;
        for each cycle c in formerCycles do
            let {m, ..., n} be the periods covered by c;
            if no decision variable P̃_m, ..., P̃_n is assigned then
                condition ← false;
        if condition then
            if Eq. 3.53 for period t in cycle and former replenishment
            cycles in formerCycles is not satisfied then
                backtrack();
        counter ← counter + 1;
end
```

Since we know the probability mass function $f(\cdot)$ for each period in the planning horizon we can easily compute the probability $Pr(\omega_t)$ for each scenario $\omega_t \in \Omega_t$. We have four of these scenarios for each period $t \in \{3, \ldots, N\}$, since we are

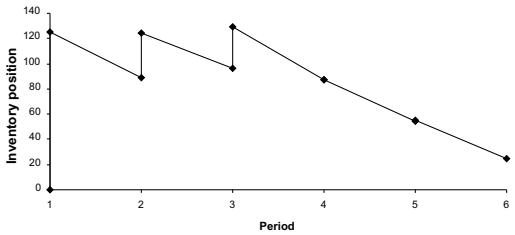


Figure 3.1: Optimal policy under stochastic lead time, $f_i(t) = \{0.3, 0.2, 0.5\}$.

Policy cost: 356					
Period (t)	1	2	3	4	5
$\tilde{d}_t$	36	28	42	33	30
R_t	125	124	129	87	55
δ_t	1	1	1	1	1
Shortage probability	–	–	5%	5%	5%

Table 3.1: Optimal solution.

placing an order in every period:

- S_1, $\Pr\{S_1\} = 0.15 = (0.3 + 0.2)0.3$; in this scenario at period t all the orders placed are received. That is the order placed in period $t-1$ is received immediately (probability 0.3), or after one period (probability 0.2), while the order placed in period t is received immediately (probability 0.3)

- S_2, $\Pr\{S_2\} = 0.35 = (0.3 + 0.2)(0.2 + 0.5)$; in this scenario at period t we don't receive the last order placed in period t. That is the order placed in period $t-1$ is received immediately (probability 0.3), or after one period (probability 0.2), while the order placed in period t is not received immediately, therefore it is received after one period (probability 0.2), or after two periods (probability 0.5)

- S_3, $\Pr\{S_3\} = 0.35 = 0.5(0.2 + 0.5)$; in this scenario at period t we don't receive the last two orders placed in periods t and $t-1$. That is the order placed in period $t-1$ is received after two periods (probability 0.5), and the order placed in period t is not received immediately, therefore it is received after one period (probability 0.2), or after two periods (probability 0.5)

- S_4, $\Pr\{S_4\} = 0.15 = 0.5 \cdot 0.3$; in this scenario at period t we don't receive the order placed in period $t-1$ and we observe order-crossover. That is the order placed in period $t-1$ is received after two periods (probability 0.5), and the order placed in period t is received immediately (probability 0.3)

In the described scenarios every possible configuration is considered. We do this without any loss in generality. In fact if some of the configurations are unrealistic (for instance if we assume that order-crossover may not take place) we just need to set the probability of the respective scenario to zero. Now it is possible to write condition 3.53 for each period $t \in \{3, \ldots, N\}$. Let us consider period 3:

$$\begin{aligned} \Pr\{S_1\} \cdot G\left(\frac{129-42}{0.3\sqrt{42^2}}\right) + \Pr\{S_2\} \cdot G\left(\frac{124-(28+42)}{0.3\sqrt{28^2+42^2}}\right) + \\ \Pr\{S_3\} \cdot G\left(\frac{125-(36+28+42)}{0.3\sqrt{36^2+28^2+42^2}}\right) + \\ \Pr\{S_4\} \cdot G\left(\frac{125+(129-124)-(36+42)}{0.3\sqrt{36^2+42^2}}\right) = 94.60\% \cong 95\% \end{aligned} \tag{3.65}$$

where $G(\cdot)$ is the standard normal distribution function. This means that the combined effect of order delivery delays in our policy, all possible scenarios taken into account, gives a no stock-out probability of about 95% for period 3. Let us consider period 4:

$$\begin{aligned} \Pr\{S_1\} \cdot G\left(\frac{87-33}{0.3\sqrt{33^2}}\right) + \Pr\{S_2\} \cdot G\left(\frac{129-(42+33)}{0.3\sqrt{42^2+33^2}}\right) + \\ \Pr\{S_3\} \cdot G\left(\frac{124-(28+42+33)}{0.3\sqrt{28^2+42^2+33^2}}\right) + \\ \Pr\{S_4\} \cdot G\left(\frac{124+(87-129)-(28+33)}{0.3\sqrt{28^2+33^2}}\right) = 94.89\% \cong 95\%. \end{aligned} \tag{3.66}$$

Period (t)	1	2	3	4	5	6	7	8
$\tilde{d}_t$	15	18	13	33	30	18	23	15

Table 3.2: Forecasts of period demands.

Let us consider period 5:

$$\Pr\{S_1\} \cdot G\left(\frac{55-30}{0.3\sqrt{30^2}}\right) + \Pr\{S_2\} \cdot G\left(\frac{87-(33+30)}{0.3\sqrt{33^2+30^2}}\right) + \Pr\{S_3\} \cdot G\left(\frac{129-(42+33+30)}{0.3\sqrt{42^2+33^2+30^2}}\right) + \Pr\{S_4\} \cdot G\left(\frac{129+(55-87)-(42+30)}{0.3\sqrt{42^2+30^2}}\right) = 94.53\% \cong 95\%. \tag{3.67}$$

We showed that the given solution satisfies the required service level for every period $t \in \{3, \ldots, N\}$. $\diamond$

3.7 Experiments

In this section we will solve to optimality an 8-period inventory problem under stochastic demand and lead time. Different lead time configurations are considered. The stochastic, deterministic and zero lead time cases are compared. As in the previous example we assume an initial null inventory level and a normally distributed demand with a coefficient of variation $\sigma_t/\tilde{d}_t = 0.3$ for each period $t \in \{1, \ldots, 8\}$. The expected values $\{\tilde{d}_t\}$ for the demand in each period are listed in Table 3.2. The other parameters are $a = 30$, $h = 1$, $v = 0$, $\alpha = 0.95 (z_{\alpha=0.95} = 1.645)$. Initially we consider the problem under stochastic demand and no lead time, an efficient CP approach to find policy parameters in this case was presented in [87, 92]. Obviously our approach is general and can provide solutions for this case as well, although less efficiently. The optimal solution for the instance considered is presented in Fig. 3.2, details about the optimal policy are reported in Table 3.3. We observe 5 replenishment cycles, policy parameters are: cycle lengths$= [1, 2, 1, 2, 2]$ and order-up-to-positions$= [72, 42, 49, 65, 52]$. The shortage probability is at most 5%, therefore the service level is met in ev-

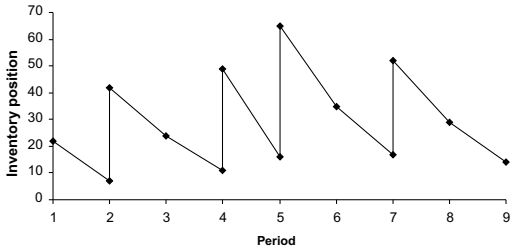


Figure 3.2: Optimal policy under no lead time.

$E\{TC\}$: 303	Average Inventory Level: 18.5							
Period (t)	1	2	3	4	5	6	7	8
R_t	22	42	24	49	65	35	52	29
δ_t	1	1	0	1	1	0	1	0
Shortage probability	5%	0%	5%	5%	0%	5%	0%	5%

Table 3.3: Optimal policy under no lead time.

ery period. The $E\{TC\}$ is 303 and the average inventory level for the policy, computed by simulating demands and lead times according to the given probability distribution function and probability mass function respectively, is 18.5 units. Since we will consider a lead time of at most 2 periods in our examples, in order to make comparisons meaningful between different instances, for the deterministic lead time cases we computed the average inventory level over 6 periods starting from period $L + 1$, where L is the lead time length, for the stochastic lead time cases we computed again the average inventory level over 6 periods, but starting from period $\tilde{L} + 1$, where $\tilde{L}$ is the average lead time length.

We now consider the same instance, but with a deterministic lead time of one period. The optimal solution is presented in Fig. 3.3, details about the optimal policy are reported in Table 3.4. We observe now only 4 replenishment cycles, policy parameters are: cycle lengths$= [2, 1, 2, 3]$ and order-up-to-positions$= [59, 64, 105, 72]$. Again the shortage probability is at most 5% in every period, which means that the service level constraint is met. The $E\{TC\}$ is 456 and the average inventory level for the policy is 25.7 units. Therefore we observe now an expected total cost that is 50.5% higher than the zero lead time case. The replenishment plan is significantly affected by the lead time both in term of re-

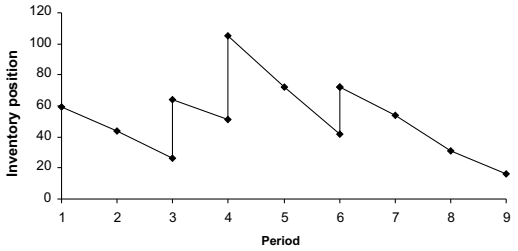


Figure 3.3: Optimal policy under deterministic one period lead time.

$E\{TC\}$: 456	Average Inventory Level: 25.7							
Period (t)	1	2	3	4	5	6	7	8
R_t	59	44	64	105	72	72	54	31
δ_t	1	0	1	1	0	1	0	0
Shortage probability	–	0%	5%	5%	0%	5%	0%	5%

Table 3.4: Optimal policy under deterministic one period lead time, notice that the service level in the first period can obviously not be controlled.

plenishment cycle lengths and order-up-to-positions. The average inventory level observed is higher than the one in the zero lead time case.

When a deterministic lead time of two periods is considered, as the reader may expect, we observe again higher costs and a different replenishment policy. The optimal solution is presented in Fig. 3.4, details about the optimal policy are reported in Table 3.5. The number of replenishment cycles is now again 5, policy parameters are: cycle lengths$= [1, 1, 2, 1, 3]$ and order-up-to-positions$= [59, 84, 119, 92, 72]$. The service level constraint is met in every period. The $E\{TC\}$ is 602 and the average inventory level for the policy is 23.2

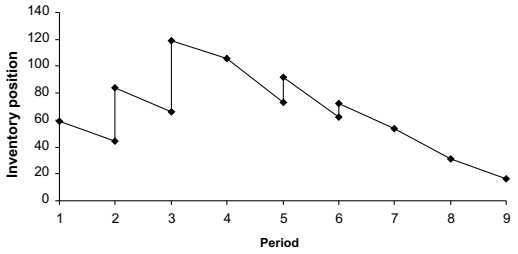


Figure 3.4: Optimal policy under deterministic two periods lead time.

$E\{TC\}$: 602	Average Inventory Level: 23.2							
Period (t)	1	2	3	4	5	6	7	8
R_t	59	84	119	106	92	72	54	31
δ_t	1	1	1	0	1	1	0	0
Shortage probability	–	–	5%	5%	0%	5%	5%	5%

Table 3.5: Optimal policy under deterministic two periods lead time.

Lead Time	$\tilde{I}$	$E_{\tilde{I}}\{TC\}$
0	18.5	261.0
1	25.7	274.2
2	23.2	289.2

Table 3.6: Deterministic lead time. Average inventory levels and respective expected total cost.

units. This means that we observe a cost 98.6% and 32.0% higher than respectively the zero lead time case and the one period lead time case. The replenishment plan is again completely modified as a consequence of the lead time length. The average inventory level observed is slightly lower than in the former cases. This is due to the fact that in this replenishment plan we schedule 5 orders, while in the optimal replenishment plan under a deterministic lead time of one period only 4 orders are planned.

In Table 3.6 we report the expected total cost $E_{\tilde{I}}\{TC\}$ computed with respect to the average inventory level $\tilde{I}$ for the three cases presented so far.

We now concentrate on two instances where a stochastic lead time is considered and we compare results with the former cases. Firstly we analyze a stochastic lead time with probability mass function $f_i(t) = \{0.2(0), 0.6(1), 0.2(2)\}$. That is an order is received immediately with probability 0.2, after one period with probability 0.6, and after two periods with probability 0.2. The optimal solution is presented in Fig. 3.5, details about the optimal policy are reported in Table 3.7. The number of replenishment cycles is again 5 as in the two period lead time case, policy parameters are: cycle lengths$= [1, 1, 2, 1, 3]$ and order-up-to-positions$= [50, 72, 101, 79, 72]$. Therefore we see that the number and the length of replenishment cycles does not change from the deterministic two period lead

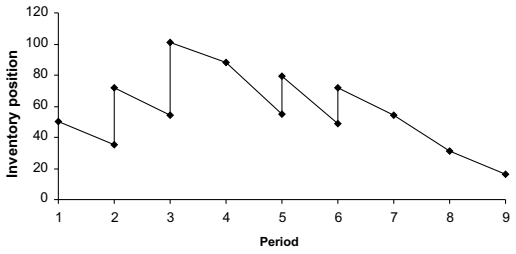


Figure 3.5: Optimal policy under stochastic lead time, $f_i(t) = \{0.2(0), 0.6(1), 0.2(2)\}$.

$E\{TC\}$: 532	Average Inventory Level: 32.8							
Period (t)	1	2	3	4	5	6	7	8
R_t	50	72	101	88	79	72	54	31
δ_t	1	1	1	0	1	1	0	0
Shortage probability	–	–	5%	5%	3%	5%	5%	5%

Table 3.7: Optimal policy under stochastic lead time, $f_i(t) = \{0.2(0), 0.6(1), 0.2(2)\}$, in periods $\{1, 2\}$ the inventory cannot be controlled.

time case, although we observe lower order-up-to-positions as we may expect since the lead time is in average one period therefore lower than in the former case. Also the cost reflects this, in fact it is 11.6% lower than in the two period deterministic lead time case. On the other hand we observed an average inventory level of 32.8, obviously affected by the uncertainty now associated with the lead time. It should be noted that the uncertainty of the lead time plays a significant role, in fact although the average lead time is one period, the structure of the policy resembles much more the one under a two period deterministic lead time than the one under a deterministic one period lead time. Moreover the expected total cost is 16.6% higher than in this latter case.

We finally consider a different probability mass function for the lead time: $f_i(t) = \{0.5(0), 0.0(1), 0.5(2)\}$, which means that we maintain the same average lead time of one period, but we increase its variance. The optimal solution is presented in Fig. 3.6, details about the optimal policy are reported in Table 3.8. The number of replenishment cycles is still 5, policy parameters are: cycle lengths$= [1, 1, 2, 1, 3]$ and order-up-to-positions$= [50, 72, 101, 79, 72]$. Although the average lead time is still one period, order-up-to-positions are slightly higher

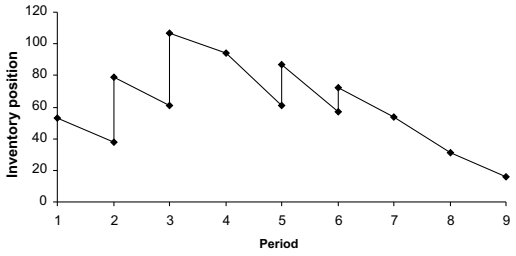


Figure 3.6: Optimal policy under stochastic lead time, $f_i(t) = \{0.5(0), 0.0(1), 0.5(2)\}$.

$E\{TC\}$: 562	Average Inventory Level: 35.5							
Period (t)	1	2	3	4	5	6	7	8
R_t	53	79	107	94	87	72	54	31
δ_t	1	1	1	0	1	1	0	0
Shortage probability	–	–	5%	5%	0%	5%	5%	5%

Table 3.8: Optimal policy under stochastic lead time, $f_i(t) = \{0.5(0), 0.0(1), 0.5(2)\}$.

Lead Time	$\tilde{I}$	$E_{\tilde{I}}\{TC\}$
$f_i(t) = \{0.2(0), 0.6(1), 0.2(2)\}$	32.8	346.8
$f_i(t) = \{0.5(0), 0.0(1), 0.5(2)\}$	35.5	363.0

Table 3.9: Stochastic lead time. Average inventory levels and respective expected total cost.

than in the former case where the variance of the lead time was lower. Also the cost reflects this, in fact it is 5.6% higher than in the former case, but still lower than the expected total cost of the two period deterministic lead time case. Moreover we observed an average inventory level of 35.5, again affected by the uncertainty associated with the lead time.

In Table 3.9 we report the expected total cost $E_{\tilde{I}}\{TC\}$ computed with respect to the average inventory level $\tilde{I}$ for the two cases where the lead time is stochastic.

To summarize, in our experiments we saw that supplier lead time uncertainty may significantly affect the structure of the optimal (R^n,S^n) policy. Computing optimal policy parameters constitutes a hard computational and theoretical

challenge. Under different degrees of lead time uncertainty, when other input parameters for the problem remain fixed, order-up-to-positions and reorder points in the optimal policy change significantly. Realizing what the optimal decisions are for certain input parameters is a counterintuitive task. Our approach provides a systematic way to compute these optimal policy parameters.

3.7.1 Analyzing the cost associated with a set of optimal policy parameters

From the experiments presented interesting insights can be obtained by observing the behavior of the expected total cost and of the average inventory level for different lead time configurations. Let us firstly observe how the expected total cost changes when the lead time changes. For a deterministic lead time, as we increase its value, the cost increases significantly when the objective function considers the expected inventory position. Intuitively this is due to the fact that every replenishment cycle covering periods $i, \ldots, j$ has to cope not only with the uncertainty associated with periods $i, \ldots, j$, but also with the variability of the demand over $j+1, \ldots, j+L-1$, where L is the lead time length. In fact the order placed in period $j+1$ will be received only after L periods. When the expected inventory level is considered, the increase ratio is lower, since we only pay the cost of the uncertainty associated with the increased buffers and we do not charge holding cost on the outstanding orders. When the lead time is stochastic and the expected inventory position is considered, the optimal policy cost is affected by the expected value of the lead time and by its variability. In fact in the last two examples presented the stochastic lead time has the same expected value of one period, but in the second example the variability is obviously higher. This directly translates into a cost difference where the lead time with probability mass function $\{0.5(0), 0(1), 0.5(2)\}$ results 5.6% more costly than the one with probability mass function $\{0.2(0), 0.6(1), 0.2(2)\}$. Nevertheless in both the cases the cost observed is lower than the one observed when the lead time is deterministic and its value is two. This can be explained by the fact that the buffers required to guarantee a given service level under a deterministic two period lead time represent a worst case scenario for every instance where the lead time is stochastic and its length

can be at most two periods. More formally this directly follows from Eq. 3.24, which determines the minimum expected inventory position required at the end of each replenishment cycle to guarantee the given service level. Although, when holding cost is charged on the expected inventory position, the behavior of the expected total cost is quite intuitive and it easily follows from the formulas presented, a dedicated reasoning must be given to explain the behavior of the average inventory level and of the expected total cost when holding cost is charged on the expected inventory level.

In the examples presented the reader may observe that a stochastic lead time distributed as follows, $\{0.2(0), 0.6(1), 0.2(2)\}$, produces an expected total cost $E\{TC\}$ lower than the one produced by a deterministic lead time of two periods. In contrast, the average inventory level $\tilde{I}$ — as well as the respective expected total cost $E_{\tilde{I}}\{TC\}$ — associated with the optimal policy computed for such a stochastic lead time is higher than the one obtained for a deterministic lead time of two periods. The reason for this is that, when we consider the expected inventory level, under a deterministic lead time we keep high buffer stocks, but we do not charge holding cost on outstanding orders, therefore the impact on the holding cost will be limited to the increase in the required buffer stocks. Under a stochastic lead time, the expected inventory level is affected by the increased buffer stocks in a similar manner, but it is also directly affected by the lead time expected value and by its variability. In fact, whenever an order has associated a short lead time, this will produce a high inventory level carried over to next periods. These scenarios may obviously affect the average inventory level of the optimal policy, while their effect on the expected inventory position is limited to the increased buffer stock levels, since the holding cost in this case is always charged also on outstanding orders. For instance a stochastic lead time distributed as follows, $\{0.5(0), 0(1), 0.5(2)\}$, produces the highest average inventory level — and expected total cost $E_{\tilde{I}}\{TC\}$ — among all the instances we considered in our set of examples. This can be explained by noticing that under a more variable lead time we will keep higher buffer stocks, and often, when the realized lead time is low, a high inventory level is accumulated and carried over to next periods before being consumed by the demand.

In conclusion we emphasize that, given a certain lead time (deterministic or

stochastic), it may be relevant for certain firms to optimize the holding cost on the expected inventory position rather than on the expected inventory level. Nevertheless if we are interested in comparing the optimal policy cost for different lead time lengths and lead time probability mass functions, then we should note that the costs obtained with these two formulations do not follow the same trend, and it is necessary to compare optimal costs obtained with the specific formulation we wish to analyze. For instance if we optimize in terms of the expected inventory position ($E\{TC\}$) the instance with a deterministic lead time of two periods and the one with a stochastic lead time distributed as follows, $\{0.5(0), 0(1), 0.5(2)\}$, our model suggests that a deterministic lead time of two periods is more costly. In contrast, since both the optimal policies place the same number of orders, by analyzing the average inventory level computed for the two instances, it is easy to notice that, when the cost is computed with respect to the expected inventory level ($E_{\tilde{I}}\{TC\}$), then the stochastic lead time results more costly.

3.8 Conclusions

A novel approach to compute (R^n,S^n) policy parameters under stochastic lead time has been presented. We have also showed how to model such a problem when a dynamic deterministic lead time is considered. The assumptions under which we developed our approach for the stochastic lead time case proved to be less restrictive than those commonly adopted in the literature for complete methods. In particular we faced the problem of order-crossover, which is a very active research topic as Riezebos show in [68] and [69]. Our approach merged well known concepts such as deterministic equivalent modeling of chance-constraints [18] and scenario based approach [91] in order to produce an effective way of solving (R^n,S^n) policy under stochastic lead time. Since we are employing CP to implement our approach we may benefit from special purpose constraint propagation techniques and cost based filtering methods that can certainly speed up the search process. Therefore in our future research we aim to develop specific filtering algorithms able to significantly speed up the search for the optimal (R^n,S^n) policy parameters under stochastic lead time.

Chapter 4

Paper III: Cost-based filtering for stochastic constraint programming

R. Rossi, S. A. Tarim, B. Hnich and S. Prestwich

Abstract

Cost-based filtering is a novel approach that combines techniques from Operations Research (OR) and Constraint Programming (CP) to filter from decision variable domains values that do not lead to better solutions [32]. Stochastic Constraint Programming is a framework for modeling combinatorial optimization problems that involve uncertainty [98]. In this work, we show how to perform cost-based filtering for certain classes of stochastic constraint programs. Our approach is based on a set of known inequalities borrowed from stochastic programming — a branch of OR concerned with modeling and solving problems involving uncertainty. We discuss bound generation and cost-based domain filtering procedures for a well known problem in the stochastic programming literature, the static stochastic knapsack problem. We also apply our technique to a stochastic sequencing problem. Our results clearly show the value of the proposed approach over a pure scenario based stochastic constraint programming formulation both in terms of explored nodes and run times.

4.1 Introduction

Constraint Programming (CP) [1] has been recognized as a powerful tool for modeling and solving combinatorial optimization problems. CP provides global constraints offering concise and declarative modeling capabilities and efficient domain filtering algorithms. These algorithms remove combinations of values which cannot appear in any consistent solution. Cost-based filtering is an elegant way of combining techniques from CP and Operations Research (OR) [32]. OR-based optimization techniques are used to remove from variable domains values that cannot lead to better solutions. This type of domain filtering can be combined with the usual CP-based filtering methods and branching heuristics, yielding powerful hybrid search algorithms. Cost-based filtering is a novel technique that has been the subject of significant recent research.

Stochastic constraint programming (SCP) [98] is an extension of CP, in which there is a distinction between decision variables, which we are free to set, and stochastic (or observed) variables, which follow some probability distribution. SCP is meant to deal with problems where uncertainty comes into play. Uncertainty may take different forms: data about events in the past may not be known exactly due to measuring or difficulties in sampling, data about events in the future may simply not be known with certainty.

In this work we propose a novel approach to perform cost-based filtering for certain classes of stochastic constraint programs. Our approach is based on a well known inequality borrowed from stochastic programming [11], a branch of OR that is concerned with modeling constraint satisfaction/optimization problems under uncertainty. We implemented this approach for two problems in which uncertainty plays a role. In both cases we obtained significant improvements with respect to a pure stochastic constraint programming formulation both in terms of explored nodes and run-times.

The rest of the paper is structured as follows. In Section 4.2 we give the necessary formal background. In Section 4.3 we review relevant inequalities for stochastic programming. In Section 4.4, we introduce global optimization chance constraints. We describe our empirical results in Section 4.5 and review related works in Section 4.6. We conclude and outline our future work in Section 4.7.

4.2 Formal Background

A *Constraint Satisfaction Problem* (CSP) [1] is a triple $\langle V, C, D\rangle$, where $V = \{V_1, \ldots, V_n\}$ is a set of decision variables, D is a function mapping each element of V to a domain of potential values, and C is a set of constraints stating allowed combinations of values for subsets of variables in V. A *solution* to a CSP is an assignment to every variable of a value in its domain, such that all of the constraints are satisfied. We may also be interested in finding a feasible solution that maximizes (minimizes) the value of a given objective function over a subset of the variables. With no loss of generality, we restrict our discussion to maximization problems.

Optimization-oriented global constraints embed an optimization component, representing a proper relaxation of the constraint itself, into a global constraint [32]. This component provides three pieces of information: (a) the optimal solution of the relaxed problem; (b) the optimal value of this solution representing an upper bound on the original problem objective function; (c) a *gradient function* **grad(**V**,**v**)**, which returns for each couple variable-value (V,v) an optimistic evaluation of the profit obtained if v is assigned to V. These pieces of information are exploited both for propagation purposes and for guiding the search.

In [98], a *stochastic CSP* is defined as a 6-tuple $\langle V, S, D, P, C, \theta\rangle$, where V is a set of decision variables and S is a set of stochastic variables, D is a function mapping each element of V and each element of S to a domain of potential values. A decision variable in V is *assigned* a value from its domain. P is a function mapping each element of S to a probability distribution for its associated domain. C is a set of constraints. A constraint $h \in C$ that constrains at least one variable in S is a *chance-constraint*. θ_h is a threshold value in the interval $[0, 1]$, indicating the minimum satisfaction probability for chance-constraint h. Note that a chance-constraint with a threshold of 1 (or without any explicit threshold specified) is equivalent to a hard constraint. A stochastic CSP consists of a number of *decision stages*. A decision stage is a pair $\langle V_i, S_i\rangle$, where V_i is a set of decision variables and S_i is a set of stochastic variables. In an m-stage stochastic CSP, V and S are partitioned into disjoint sets, $V_1, \ldots, V_m$ and $S_1, \ldots, S_m$, and we consider multiple stages, $\langle V_1, S_1\rangle, \langle V_2, S_2\rangle, \ldots, \langle V_m, S_m\rangle$. To solve an m-stage stochastic CSP

an assignment to the variables in V_1 must be found such that, given random values for S_1, assignments can be found for V_2 such that, given random values for S_2, ..., assignments can be found for V_m so that, given random values for S_m, the hard constraints are satisfied and the chance constraints are satisfied in the specified fraction of all possible scenarios. The solution of an m-stage stochastic CSP is represented by means of a *policy tree* [91]. A policy tree is a set of decisions where each path represents a different possible scenario and the values assigned to decision variables in this scenario. Let $\mathcal{S}$ denote the space of policy trees representing all the solutions of a stochastic CSP. We may be interested in finding a feasible solution, i.e. a policy tree $s \in \mathcal{S}$, that maximizes the value of a given objective function $f(\cdot)$ over the stochastic variables S (edges of the policy tree) and over a subset $\widehat{V} \subseteq V$ of the decision variables (nodes in the policy tree). A *Stochastic COP* is then defined in general as $\max_{s \in \mathcal{S}} f(s)$. In [98] a policy based view of stochastic constraint programs is proposed. Such an approach has been further investigated in [5]. An alternative semantics for stochastic constraint programs comes from a scenario-based view [11, 91]: this solution method consists in generating a scenario-tree that incorporates all possible realizations of discrete stochastic variables into the model explicitly.

4.3 Value of Stochastic Solutions

Let Ξ be a discrete stochastic (vector) variable whose realizations correspond to the various scenarios. Recall that in the policy based view of stochastic CP a scenario is a set of edges in the policy tree connecting the root to a leaf. Define

$$\mathrm{P} = \max_{x \in S} z(x, \xi)$$

as the optimization problem associated to one particular scenario $\xi \in \Xi$, where S is a *finite* set, and $z(x, \xi)$ is a real valued function of two (vector) variables x and ξ. Note that in what follows the discussion is dual for minimization problems. In order to simplify the notation used we will here use the same notation for referring to a problem and to the value of its optimal solution. The one or the other meaning will be made clear by the context.

The function $z(x, \xi)$ can be seen as a payoff table that for a given decision x provides the profit with respect to a given scenario ξ having probability $\Pr\{\xi\}$. We may be then interested in computing the optimal solution value to the *recourse problem* [11] RP(P)$= \max_{x \in S} \sum_{\Xi} \Pr\{\xi\} z(x, \xi)$. This can be expressed, by using the expectation operator $\mathbb{E}$, as

$$\mathrm{RP(P)} = \max_{x \in S} \mathbb{E} z(x, \Xi),$$

with an optimal solution x^*.

The *expected value problem*, the deterministic problem obtained by replacing all the stochastic (vector) variables by their expected values, is defined as

$$\mathrm{EV(P)} = \max_{x \in S} z(x, \mathbb{E}[\Xi]).$$

Let us denote by $\widehat{x}$ an optimal solution of the expected value problem, called the *expected value solution*. Anyone familiar with stochastic programming or realizing that uncertainty is a fact of life would feel a little insecure about taking decision $\widehat{x}$. Indeed, unless such a decision is independent of Ξ, there is no reason to believe that this decision is in any way close to the optimal solution of the recourse problem.

For any stochastic maximization (minimization) program, under the assumptions that (i) $z(x, \Xi)$, the profit function, is a concave[†] (convex) function of Ξ and (ii) $\max_{x \in S} z(x, \Xi)$ ($\min_{x \in S} z(x, \Xi)$) exists for all Ξ,

PROPOSITION 1. *EV(P) - RP(P)* ≥ 0 *(EV(P) - RP(P)* ≤ 0*).*

Proof. A proof is given in [4]. □

It directly follows that EV(P) $\geq$ RP(P) (EV(P) $\leq$ RP(P)). On this inequality we will base our cost-based filtering strategies.[‡] Assumption (i) restricts the form of the cost function. Many real life applications exhibit such a behavior in the profit

[†] A real-valued function f is *convex* if for any x_1, x_2 in the domain and any $\lambda \in [0, 1]$, $\lambda f(x_1) + (1 - \lambda) f(x_2) \geq f(\lambda x_1 + (1 - \lambda) x_2)$ [16]. f is *concave* if $-f$ is convex.

[‡] Other inequalities are discussed in [11], pp. 140–141. Effective relaxations can be also built on these other inequalities.

(cost) function. Nevertheless, often it is possible to encounter stochastic constraint programs whose objective exhibits a generalized non-convex dependence on the stochastic variables. Note that, although the classical Jensen (Proposition 1) and Edmundson-Madansky type bounds [11], which we will employ in the following sections, or their extensions are generally not available for such problems, tight bounds may still be constructed under mild regularity conditions as discussed in [57]. Assumption (ii) states that Proposition 1 can be applied only when a feasible solution exists and its existence is not affected by the distribution of the stochastic variables. As suggested in [11], also this assumption is realistic. In fact, in stochastic programs people usually tend to associate a high cost, rather than an infeasibility to decisions that are poor with respect to the random outcomes. Assumption (ii) is typically not respected in problems where chance-constraints appear. We will not discuss how to handle generic chance-constraints and how to produce deterministic equivalent reformulations for them in EV(P), the reader may refer to [19]. In this work we will consider only examples on stochastic COPs that satisfy assumptions (i) and (ii). In particular, to comply with assumption (ii), we will consider problems for which a feasible solution always exists and for which the chance-constraints are "hard" ($\theta = 1$). Note that "hard" chance-constraints in RP(P) become deterministic in EV(P).

4.4 Global optimization chance-constraints

Solving stochastic constraint programs is computationally a challenging task. In [98], the computational complexity — membership in PSPACE — of these models is discussed. In [91], the authors proposed a standard way of compiling down these models into conventional (non-stochastic) constraint programming models that can be solved by any available commercial software. This approach employs a scenario-based [11] modelling strategy for representing stochastic variables. Of course this approach has a price since the number of scenarios that need to be considered in order to fully represent the problem grows exponentially with the number of decision stages in the problem. A possible way to overcome this difficulty is to reduce the number of scenarios considered by sampling them, but this obviously affects the completeness of the model. Another possibility con-

sists instead in developing specialized and efficient filtering strategies. For this purpose *global chance-constraints* have been proposed in [75]. These constraints differ from conventional global constraints in the fact that they represent relations among a non-fixed number of decision variables and stochastic variables.

In this work, by creating a parallel with [32], we present *optimization-oriented global chance-constraints* as a way to enhance the solving process of stochastic constraint programs. Conventional optimization-oriented global constraints perform cost-based filtering by encapsulating in global constraints optimization components representing suitable relaxations of the constraint itself. Similarly optimization-oriented global chance-constraints also encapsulate suitable relaxations of the constraint considered, but in contrast to conventional optimization-oriented global constraints this relaxation may involve stochastic variables.

A *global optimization chance-constraint* provides the same three pieces of information provided by optimization-oriented global constraints. What differs is the fact that in a global optimization chance-constraint we find two stages of relaxations. At the first stage of relaxation, we are mainly involved with the stochastic variables and we exploit well known inequalities such as the one in Proposition 1 to replace stochastic variables in our stochastic programs with deterministic quantities and to yield a valid relaxation that is a deterministic problem. This deterministic problem, however, may still be computationally very challenging (NP-Hard in general). Therefore, a second stage of relaxation may be needed to produce a further relaxation that is computationally more tractable. Finally, as we will see, a global optimization chance-constraint may also provide a valid, and possibly good, solution at each node of the search tree.

In this section and in the following ones we will refer to a running example and we will employ the following problem to better understand the concepts explained. Consider the *Static Stochastic Knapsack Problem* (SSKP) [56]: a subset of k items has to be chosen, given a knapsack of size q into which to fit the items. Each item i has an expected reward of r_i. The size $\mathcal{W}_i$ of each item is not known at the time the decision has to be made, but we assume that the decision maker has an estimate of the probability distribution of $\overline{\mathcal{W}} = (\mathcal{W}_1, \ldots, \mathcal{W}_k)$. A per unit penalty of c has to be paid for exceeding the capacity of the knapsack. By modeling this problem as a one-stage Stochastic COP, the recourse problem RP(SSKP) can be

Objective:

$$\max \left\{ \sum_{i=1}^{k} r_i X_i - c\mathbb{E}\left[\sum_{i=1}^{k} \mathcal{W}_i X_i - q \right]^+ \right\}$$

Decision variables:

(1) $X_i \in \{0,1\}$ $\quad \forall i \in 1,\ldots,k$

Stochastic variables:

$\mathcal{W}_i \rightarrow$ item i weight

Figure 4.1: RP(SSKP). Note that $[y]^+ = \max\{y, 0\}$. $\mathbb{E}$ denotes the expectation operator.

formulated as shown in Fig. 4.1. The objective function maximizes the trade off between the reward brought by the objects selected in the knapsack (those for which the binary decision variable X_i is set to 1) and the expected penalty paid for buying additional capacity units in those scenarios where the low cost capacity q is not sufficient.

Example 4.4.1. Consider 5 items, item rewards r_i are $\{10, 15, 20, 5, 25\}$. The discrete probability distribution functions $f(i)$ for the weight of item $i = 1,\ldots,5$ are respectively, $f(1) = \{10(0.5), 8(0.5)\}$, $f(2) = \{10(0.5), 12(0.5)\}$, $f(3) = \{9(0.5), 13(0.5)\}$, $f(4) = \{4(0.5), 6(0.5)\}$, $f(5) = \{12(0.5), 15(0.5)\}$. The figures in parenthesis represent the probability that an item takes a certain weight. The other problem parameters are $c = 2$, $q = 30$. The optimal solution of the recourse problem selects items $\{2, 3, 5\}$ and has a value of RP(SSKP)=49. $\diamond$

This solution can be obtained by solving a deterministic equivalent conventional constraint program obtained by employing a scenario based representation [91]. Let $\mathcal{W}_i^j$ be the realized weight of object i in scenario j. We hand-crafted a deterministic equivalent model DetEquiv(RP(SSKP)) for RP(SSKP) following the guidelines in [91]. This model is shown in Fig. 4.2. Constraint (1) states that Z_j, total excess weight in scenario j, must be greater than the sum of the weights of the objects selected in this scenario minus the low cost capacity q. Constraint (2) declares the decision variables X_i's. X_i is equal to 1 iff item i is selected in the knapsack. Constraint (3) fixes an upper bound for Z_j; this upper bound is the sum of the weights of all the k objects in scenario j. The objective function maximizes the trade off between the total reward brought by the objects selected and the sum of penalty costs — weighted by the respective scenario probability — paid for

Objective:		
$\max \left\{ \sum_{i=0}^{k} r_i X_i - c \left[\sum_{j=1}^{n} Z_j \Pr\{j\} \right] \right\}$		
Constraints:		
(1)	$Z_j \geq \sum_{i=1}^{k} \mathcal{W}_i^j X_i - q$	$\forall j \in 1, \ldots, n$
Decision variables:		
(2)	$X_i \in \{0, 1\}$	$\forall i \in 1, \ldots, k$
(3)	$Z_j \in [0, \sum_{i=1}^{k} \mathcal{W}_i^j]$	$\forall j \in 1, \ldots, n$

Figure 4.2: DetEquiv(RP(SSKP)). $\Pr\{j\}$ is the probability of scenario $j \in \{1, \ldots, n\}$. Note that $\sum_{j=1}^{n} \Pr\{j\} = 1$.

those scenarios where the low cost capacity q is not sufficient.

4.4.1 Expectation-based relaxation for stochastic variables

The first step in our cost-based filtering strategy consists in applying a relaxation involving the stochastic variables. By applying Proposition 1, if the profit (respectively cost for minimization problems) function satisfies the two assumptions discussed, an upper (lower) bound for the cost of an optimal solution to RP(P) can be obtained by solving EV(P), that is the deterministic problem where all the stochastic variables are replaced by their respective expected values.

Lemma 4.4.1. *The profit function for $RP(SSKP)$ is concave in $\overline{\mathcal{W}}$.*

Proof. When proving concavity w.r.t. $\overline{\mathcal{W}}$ we can ignore the constant term $\sum_{i=1}^{k} r_i X_i$. What remains is $f(\overline{\mathcal{W}}) = -c\mathbb{E}\left[\overline{W}^T \cdot \overline{X} - q\right]^+$, where "$\cdot$" is the inner product and $\overline{\mathcal{W}}^T$ is vector $\overline{\mathcal{W}}$ transposed. We now prove that $-f(\overline{\mathcal{W}}) = c\mathbb{E}\left[\overline{\mathcal{W}}^T \cdot \overline{X} - q\right]^+$ is convex in $\overline{\mathcal{W}}$. By recalling that a maximum of convex functions is convex [16], this function is clearly convex w.r.t. each element of vector $\overline{\mathcal{W}}$ and it is therefore convex in $\overline{\mathcal{W}}$. This implies that $-f$ is concave in $\overline{\mathcal{W}}$. □

Obviously, in RP(SSKP), it is always possible to find a feasible assignment for decision variables, therefore both the assumptions are satisfied for this problem. The expected value problem EV(SSKP) can be obtained by replacing every random variable $\mathcal{W}_i$ in RP(SSKP) with the respective expected value $\mathbb{E}[\mathcal{W}_i]$, thus obtaining a fully deterministic model.

Example 4.4.2. We here solve the problem where the weights of the objects are deterministic and equal to the respective expected weights[§]: $\lfloor \mathbb{E}[f(1)] \rfloor = 9$, $\lfloor \mathbb{E}[f(2)] \rfloor = 11$, $\lfloor \mathbb{E}[f(3)] \rfloor = 11$, $\lfloor \mathbb{E}[f(4)] \rfloor = 5$, $\lfloor \mathbb{E}[f(5)] \rfloor = 13$. This problem provides the first two pieces of information needed by our cost-based filtering method, that is (a) the optimal solution of the relaxed problem and (b) the optimal value of this solution, which represents, according to Proposition 1, an upper bound for the original problem objective function. In our running example this solution selects items $3, 4, 5$ and has a value of EV(SSKP)$= 50$. $\diamond$

4.4.2 Relaxing the expected value problem

It should be noted that, although the expected value problem is easier than the recourse problem, it may still be difficult to solve (NP-Hard). For this reason we can further relax the expected value problem in order to obtain a valid bound by solving an easier problem. Let R(EV(P)) be a generic relaxation of EV(P), then in a maximization problem EV(P)$\leq$ R(EV(P)), therefore R(EV(P)) provides a valid bound for the recourse problem.

In SSKP, for instance, instead of solving to optimality the deterministic (NP-Complete) knapsack problem obtained for the expected value scenario, we may instead solve in linear time its continuous relaxation, thus obtaining Dantzig's upper bound, DUB(EV(SSKP)), for it [63]. DUB(EV(SSKP)) $\geq$ EV(SSKP) and therefore DUB(EV(SSKP)) $\geq$ RP(SSKP). DUB(EV(SSKP)) is a valid upper bound for our recourse problem.

Example 4.4.3. To obtain DUB(EV(SSKP)) we order items for profit over expected weight: $\{25/13, 20/11, 15/11, 10/9, 5/5\}$, and we insert items until the first that does not fit completely into the remaining knapsack capacity. Of this last item we take a fraction of the profit proportional to the capacity available. Therefore DUB(EV(SSKP))$= 25 + 20 + (6 * 15/11) = 53.18$. $\diamond$

Obviously now at any node of the search tree it is possible to solve the expected value problem taking into account decision variables already assigned and

[§]Since the problem is here a maximization one, the expected weight of each object is rounded down to the nearest integer ($\lfloor\ \rfloor$) in order to keep optimistic the bound provided by the relaxation.

exploit this new bound obtained in order to exclude part of the tree that cannot lead to a better solution.

In [32] the authors discuss filtering strategies based on reduced costs (RC). As we shall see in the next section a similar technique can be adopted for SSKP as well, provided that an efficient way of obtaining bounds is available for the expected value problem.

4.4.3 Cost-based filtering

In order to perform cost-based filtering, as in RC-based filtering, we need a *gradient function* **grad(**V**,**v**)**, which returns for each couple variable-value (V,v) an optimistic evaluation of the profit obtained if v is assigned to V.

This function is obviously problem dependent, but regardless of the strategy adopted in the former section — i.e. whenever we are using a relaxation for the expected value problem or we are solving this problem to optimality — it is possible to specify it and use it to filter provably suboptimal values.

In what follows we present a gradient function for SSKP. At each node of the search tree, in order to compute this function, we use a continuous relaxation on the expected value problem similar to the one proposed by Dantzig for the well known 0-1 Knapsack Problem [63]. We will now define the gradient function for SSKP by reasoning on the expected value problem. Assume that a partial assignment for decision variables is given. Let K be the set of all the items in the problem, $|K| = k$. Let S be the set of items for which a decision has been fixed, with $|S| < k$. Let q^* be the sum of the expected weights of the elements in S that are part of the knapsack. The profit $\overline{r}$ associated to this assignment is equal to the sum of the profits of the items in the knapsack minus the eventual expected penalty cost $c(q^* - q)$, if $q - q^*$ is negative. Now we consider an element $i \in K/S$. There are two possible options: taking it or not into the knapsack. If we take it, we increase the profit by r_i minus any eventual expected penalty cost we pay if the expected residual capacity is already or becomes negative. Finally for every other element in K/S we check if the balance between its profit and the eventual expected penalty gives an overall positive profit and, if so, we include it into the knapsack. This procedure requires at most $O(k)$ steps for each element for which

a decision has not been taken yet, therefore it can be applied at each node of the search tree to compute a valid upper bound associated to a certain decision on an item, which therefore may be filtered if suboptimal.

Example 4.4.4. We now consider the case in which items 2 and 3 have been selected in the knapsack and item 4 is not selected. We still have to decide on items 1 and 5. The total capacity used is $c^* = 11 + 11 = 22$. The profit $\overline{r}$ brought by items 2 and 3 is 35. We consider the set of the remaining items for which a decision must be taken, $K/S \equiv \{1, 5\}$. Let us reason on item 1: this is a critical item, in fact if taken in the knapsack it will use more capacity than the residual $30 - 22 = 8$ units. If we consider the option of taking this item, then the expected profit is $\overline{r}_1 = 10 - 2 * (30 - 22 - 9) = 8$, there is no more residual capacity and item 5 is therefore excluded in the bound computation since $25 - 4 * 13 \leq 0$. The computed bound is $35 + 8 = 43$. The reasoning is similar for item 5. If we consider the option of taking this item, then the expected profit is $\overline{r}_5 = 25 - 2 * (30 - 22 - 13) = 15$, there is no more residual capacity and item 1 is therefore excluded in the bound computation since $10 - 4 * 9 \leq 0$. The computed bound is $35 + 15 = 50$. Assume now that the current best solution has a value of 46, corresponding to a knapsack that contains elements 3, 4 and 5: then element 1 can be excluded from the knapsack. $\diamond$

Obviously, as discussed in [32] the information provided by the relaxed model (expected value problem), i.e. expected weights, gradient function etc., can be also used to define search strategies. For instance in SSKP we may branch on variables according to a decreasing profit over expected weight heuristic, or selecting the one for which the chosen gradient function gives the most promising value.

4.4.4 Finding good feasible solutions

In CP, it is critical, in order to achieve efficiency, to quickly obtain a good feasible solution so that cost-based filtering can prune provably suboptimal nodes as early as possible. In Stochastic COPs the EV(P) solution can be often used as a good starting solution in the search process. If such a solution is feasible with respect to RP(P) — in our examples assumption (ii) guarantees this — we can easily compute EEV(P), that is *the expected result of using the EV(P) solution in*

the recourse problem RP(P). Furthermore, at every node of the search tree it is possible to adopt a variable fixing strategy and compute the EV(P) solution with respect to such a node, that is the best possible EV(P) solution incorporating the partial decisions represented by the given node of the search tree. This provides a full assignment for decision variables in RP(P) at each point of the search. By using this assignment, we can again easily compute EEV(P). In this case EEV(P) is the cost of a feasible, and possibly good, solution for RP(P) incorporating the partial assignment identified by the current node explored in the search tree.

Example 4.4.5. In our SSKP example the solution of the expected value problem, EV(SSKP), selects items 3, 4 and 5 in the optimum knapsack. This solution is clearly feasible for RP(SSKP). We can therefore compute EEV(SSKP)$= 46$. This is, of course, a good lower bound for the objective function value. $\diamond$

4.5 Experimental results

In this section we report our computational experience on two one-stage stochastic COPs, the SSKP and the Stochastic Sequencing with Release Times and Deadlines (SSEQ). In our experiments we used Choco 1.2, an open source solver written in Java [58]. We ran our experiments on an Intel(R) Centrino(TM) CPU 1.50GHz with 2Gb of RAM.

4.5.1 Static Stochastic Knapsack Problem

We created a Choco CP model for DetEquiv(RP(SSKP)), and we implemented for it a global optimization chance-constraint incorporating the filtering discussed in the former sections. To recall, within this constraint at each node of the search tree the stochastic variables are replaced by their respective expected values. Then, after fixing decision variables according to the partial solution associated to the given search tree node, EV(SSKP) is solved and the bound obtained is used to prune suboptimal parts of the search tree. Furthermore cost-based filtering is performed as explained in Section 4.4.3. Finally EEV(P), the *expected result of using the EV(P) solution in the recourse problem*, is computed at each node of the search tree and used as a valid lower bound (profit of a feasible solution). In

fact RP(SSKP) satisfies assumption (ii) for Proposition 1, therefore the solution of EV(SSKP) is feasible for RP(SSKP).

In our experiments we adopted a randomly generated test bed similar to the one proposed in [56]. There are three sets of instances considered: the first set has $k = 10$, the second set has $k = 15$ and the third has $k = 20$ items. For all the instances, item random weights, W_i, from which scenarios are generated, are independent and normally distributed with probability distribution function $N(\mu_i, \sigma_i)$. The expected weights, μ_i, are generated from the uniform (20,30) distribution, and the weight standard deviations, σ_i, are generated from the uniform (5,10) distribution. Rewards r_i are generated from the uniform (10,20) distribution. The per unit penalty is $c = 4$, while the available low cost capacity is $q = 250$ for 20 items, $q = 187$ for 15 items, and $q = 125$ for 10 items. We randomly generated, using simple random sampling, sets of scenarios having different sizes: $\{100, 300, 500, 1000\}$. Scenarios are equally likely in terms of probability. The variable selection heuristic branches first on items with lower profit over expected weight ratio. The value selection tries first not to insert an item into the knapsack. In Table 4.1 we report our computational results. In all the instances considered our approach outperforms a pure SCP model in terms of explored nodes: the maximum improvement reaches a factor of 576.5. Run times are also shorter in our approach for almost all the instances. An exception is observed for the smallest instance, where the cost of filtering domains is not compensated by the payoff in terms of reduction of the search space. The maximum speed-up observed for run times reaches a factor of 90.5.

4.5.2 Stochastic sequencing with release times and deadlines

We consider a specific sequencing problem similar to the one considered by Hooker et. al [47]. Garey and Johnson [37] also mention this problem in their list of NP-Hard problems and they refer to it as "Sequencing with Release Times and Deadlines" (SSEQ). An optimization version of this scheduling problem was also described in [50]. The problem consists in finding a feasible schedule to process a set I of k orders (or jobs) using a set M of n parallel machines. Processing an order $i \in I$ can only begin after the release date r_i and must be completed at

Instance		Time		Nodes	
k	Scenarios	SCP	SCP-OO	SCP	SCP-OO
10	100	**0.4**	0.5	916	**100**
10	300	1.3	**0.5**	2630	**59**
10	500	2.4	**0.2**	4237	**8**
10	1000	7.2	**2.4**	6227	**120**
15	100	2.5	**0.3**	4577	**11**
15	300	15	**2.3**	10408	**252**
15	500	33	**1.1**	9982	**75**
15	1000	150	**6.3**	16957	**222**
20	100	70	**10**	102878	**1024**
20	300	250	**13**	85073	**953**
20	500	860	**9.5**	129715	**225**
20	1000	3200	**240**	134230	**7962**

Table 4.1: Experimental results for SSKP. Comparison between a pure SCP approach (SCP) and an SCP model enhanced with optimization-oriented global-chance constraints (SCP-OO), times are in seconds. In each line we indicated in bold the best performance in terms of run time and explored nodes.

the latest by the due date d_i. Order i can be processed on any of the machines. The processing time of order $i \in I$ on machine $m \in M$ is P_{im}. The model just described is fully deterministic, but we will now consider a generalization of this problem to the case where some inputs are uncertain. For convenience we will just consider uncertain processing times $\mathcal{P}_{im}$ for order $i \in I$ on machine $m \in M$. Instead of simply finding a feasible plan we now aim to minimize the expected total tardiness of the plan (the deterministic version of this problem is known as "Sequencing to minimize weighted tardiness" [37] and it is NP-Hard). A solution for our SSEQ problem consists in an assignment for the jobs on the machines and in a total order between jobs on the same machine. In such a plan, a job will be processed on its release date if no other previous job is still processing, or as soon as the previous job terminates. The recourse problem RP(SSEQ) can be formulated as a one-stage Stochastic COP. This is shown in Fig. 4.3.

Decision variable X_{im} takes value 1 iff job i is processed on machine m, decision variable S_{ab} takes value 1 iff job a is processed before job b. Constraints (1) and (2) enforce a total order among jobs on the same machine. Constraint (3) enforces that each job must be processed on one and only one machine. Constraint (4) states that the (stochastic) completion time, $\mathcal{C}_i$, of a job i minus its (stochastic) duration $\mathcal{P}_{im}$ on the machine on which it is processed must be greater than or equal to its release date r_i, where $\mathcal{C}_i$ is an auxiliary variable used for simplifying notation. Let $I_m \equiv \{\mathcal{J}_{1m}, \mathcal{J}_{2m}, \ldots, \mathcal{J}_{qm}\} \subseteq I$ be the ordered set of jobs assigned to

Objective:

$\min \left\{ \sum_{i=1}^{k} \mathbb{E}\left[\mathcal{C}_i - d_i\right]^+ \right\}$

Constraints:

(1)	$S_{ab} + S_{ba} \leq 1$	$\forall a, b \in 1, \ldots, k, a \neq b$
(2)	$X_{am} + X_{bm} \leq S_{ab} + S_{ba} + 1$	$\forall a, b \in 1, \ldots, k, a \neq b, \forall m \in 1, \ldots, n$
(3)	$\sum_{m=1}^{n} X_{im} = 1$	$\forall i \in 1, \ldots, k$
(4)	$\mathcal{C}_i - \sum_{m=1}^{n} \mathcal{P}_{im} X_{im} \geq r_i$	$\forall i \in 1, \ldots, k$
(5)	$S_{ab} = 1 \rightarrow \mathcal{C}_b \geq \mathcal{C}_a + \sum_{m=1}^{n} \mathcal{P}_{bm} X_{bm}$	$\forall a, b \in 1, \ldots, k, a \neq b$

Decision variables:

(6)	$X_{im} \in \{0, 1\}$	$\forall i \in 1, \ldots, k, \forall m \in 1, \ldots, n$
(7)	$S_{ab} \in \{0, 1\}$	$\forall a, b \in 1, \ldots, k, a \neq b$

Stochastic variables:

$\mathcal{P}_{im}$: processing time of job i on machine m

Auxiliary variables:

$\mathcal{C}_i$: stochastic completion time of job i.

Figure 4.3: RP(SSEQ). Note that $[y]^+ = \max\{y, 0\}$. $\mathbb{E}$ denotes the expectation operator.

machine m. $\mathcal{C}_{\mathcal{J}_{qm}}$ is defined recursively as $\mathcal{C}_{\mathcal{J}_{qm}} = \max\{r_{\mathcal{J}_{qm}}, \mathcal{C}_{\mathcal{J}_{(q-1)m}}\} + \mathcal{P}_{\mathcal{J}_{qm}m}$, and $\mathcal{C}_{\mathcal{J}_{0m}} = 0$. Constraint (5) states that if two jobs a and b are processed on the same machine and if a is processed before b, that is $S_{ab} = 1$, then the (stochastic) completion time of job a plus the (stochastic) duration of job b on the machine on which it is processed must be less or equal to the (stochastic) completion time of job b. Finally, the objective function minimizes the sum of the expected tardiness of each job. The tardiness is defined as $\max\{0, \mathcal{C}_i - d_i\}$. The cost function that has to be minimized can be easily proved to be convex in the random job durations. The expected total tardiness is in fact minimized for n machines. Job completion times on different machines are independent, therefore if we prove convexity for machine $m \in M$, then it directly follows that the cost function of the problem is also convex[¶]. The cost function for machine m can be expressed as $\mathbb{E}\left[\sum_{i \in I_m} (\mathcal{C}_i - d_i)^+\right]$.

Lemma 4.5.1. *The expected total tardiness for machine m is convex in the uncertain processing times $\mathcal{P}_{im}$.*

Proof. Maximum of convex functions is convex. $\mathcal{C}_{\mathcal{J}_{1m}} = r_{\mathcal{J}_{1m}} + \mathcal{P}_{\mathcal{J}_{1m}m}$ is convex: it follows that $\mathcal{C}_i$ for any $i \in I_m$ is convex, since function "$\max$" is a convex function. Therefore the objective function is convex. □

[¶]Note that the sum of convex functions is convex [16].

Objective:

$\min \left\{ \sum_{i=1}^{k} \sum_{v=1}^{w} \Pr\{w\} \left[C_i^v - d_i \right]^+ \right\}$

Constraints:

(1) $S_{ab} + S_{ba} \leq 1$	$\forall a, b \in 1, \ldots, k, a \neq b$
(2) $X_{am} + X_{bm} \leq S_{ab} + S_{ba} + 1$	$\forall a, b \in 1, \ldots, k, a \neq b, \forall m \in 1, \ldots, n$
(3) $\sum_{m=1}^{n} X_{im} = 1$	$\forall i \in 1, \ldots, k$
and $\forall v \in 1, \ldots, w$	
(4) $C_i^v - \sum_{m=1}^{n} \mathcal{P}_{im}^v x_{im} \geq r_i$	$\forall i \in 1, \ldots, k$
(5) $S_{ab} = 1 \rightarrow C_b^v \geq C_a^v + \sum_{m=1}^{n} \mathcal{P}_{bm}^v X_{bm}$	$\forall a, b \in 1, \ldots, k, a \neq b$
Decision variables:	
(6) $X_{im} \in \{0, 1\}$	$\forall i \in 1, \ldots, k, \forall m \in 1, \ldots, n$
(7) $S_{ab} \in \{0, 1\}$	$\forall a, b \in 1, \ldots, k, a \neq b$
(8) $C_i^v \in \{0, \max_{i=1,\ldots,k} r_i + \sum_{t=1}^{k} (\max_{m=1,\ldots,n} \pi_{tm}^v)\}$	$\forall i \in 1, \ldots, k, \forall v \in 1, \ldots, w$

Figure 4.4: DetEquiv(RP(SSEQ)). Note that $[y]^+ = \max\{y, 0\}$. $\Pr\{v\}$ is the probability of scenario $v \in \{1, \ldots, w\}$. Note that $\sum_{v=1}^{w} \Pr\{v\} = 1$.

In RP(SSEQ) a feasible solution can be found for any given set of stochastic job lengths, therefore both the assumptions are satisfied for this problem. We hand-crafted a deterministic equivalent model DetEquiv(RP(SSEQ)) shown in Fig. 4.4 for the RP(SSEQ) following the guidelines of scenario-based approach described in [91]. In this model, $\mathcal{P}_{im}^v$ is the deterministic length of job i on machine m in scenario v and C_i^v is the deterministic completion time of job i in scenario v.

Finally, as discussed for SSKP, we can obtain the expected value problem EV(SSEQ) by replacing every stochastic variable $\mathcal{P}_{im}$ in RP(SSEQ) with the respective expected value $\mathbb{E}[\mathcal{P}_{im}]$. Since all the chance-constraints in RP(SSEQ) are "hard", they are retained in EV(SSEQ) and they become deterministic.

We implemented DetEquiv(RP(SSEQ)) in Choco and we coded an optimization-oriented global chance-constraint which exploits the expected value problem both in order to generate valid bounds at each node of the search tree and to filter provably suboptimal values from decision variable domains. At each node of the search tree, we consider the associated partial assignment for decision variables X_{im} and S_{ab} and we fix decision variables in EV(SSEQ) according to it. Then we solve EV(SSEQ) with respect to the remaining decision variables that have not been assigned. This provides a lower bound for the cost of a locally optimal solution associated to the node considered. This bound can be used for pruning

Instance			Time		Nodes	
Jobs	Machines	Scenarios	SCP	SCP-OO	SCP	SCP-OO
3	2	10	**0.3**	**0.3**	203	**48**
3	2	30	1.3	**0.6**	701	**133**
3	2	50	3.2	**1.1**	927	**418**
3	2	100	12	**3.5**	1809	**838**
7	3	10	**180**	866	57688	**1723**
7	3	30	1800	**880**	186257	**5293**
7	3	50	3300	**1100**	212887	**6586**
7	3	100	14000	**1200**	277804	**8862**

Table 4.2: Experimental Results for SSEQ. Comparison between a pure SCP approach (SCP) and an SCP model enhanced with optimization-oriented global-chance constraints (SCP-OO), times are in seconds. In each line we indicated in bold the best performance in terms of run time and explored nodes.

suboptimal nodes. Furthermore at any given node, after performing variable fixing in EV(SSEQ) for every variable X_{im} and S_{ab} already assigned, all the remaining binary variables X_{im} that have not been assigned yet can be forward checked one by one by fixing the respective value to 1, by solving EV(SSEQ) with this new decision fixed, and by employing the new bound provided.

In order to generate instances for our experiments, we adopted release times, deadlines and deterministic processing times from the first two "hard" instances proposed in [47], the one with 3 jobs and 2 machines and the one with 7 jobs and 3 machines. In each scenario, we generated processing times uniformly distributed in $[1, 2 * J_{im}]$, where J_{im} is the deterministic processing time required for job i on machine m for the instance considered. We considered different number of scenarios in $\{10, 30, 50, 100\}$. Scenarios are equally likely in terms of probability. The variable selection heuristic branches first on binary decision variables. The value selection tries increasing values in the domain. In Table 4.2 we report the results observed with and without the improvement brought by our cost-based filtering approach.

It should be noted that in this case, in contrast to the approach employed for SSKP, we only relax stochastic variables and we do not employ a relaxation for the deterministic equivalent problem, which therefore remains NP-Hard. Recall that in SSKP we adopted Dantzig's relaxation to efficiently obtain a bound for the deterministic equivalent problem. A direct consequence of this is that, while in the SSKP example the improvement is significant both in terms of explored nodes and run times for all the instances, in this example the run time improvement starts

to be significant (a factor of 11.6) only for the largest instance (7 jobs and 3 machines) and for a high number of scenarios (100 scenarios). This is due to the fact that at every node of the search tree we solve a difficult problem (although far easier than the original stochastic constraint program) to obtain bounds and perform cost-based filtering. In terms of explored nodes, however, we obtain a significant improvement for every instance considered — the maximum improvement factor is of 32.3 — since the bounds generated are tight.

4.6 Related works

This paper extends the original work by Focacci et al. [32] on optimization-oriented global constraints. It also extends the original idea of global chance-constraints [75] to optimization problems. It should be noted that dedicated cost-based filtering techniques for stochastic combinatorial optimization problems have been presented in [88], but these techniques are specialized for inventory control problems, while those here presented can be applied to a wider class of stochastic constraint programs. On the other hand this work also builds on known inequalities borrowed from stochastic programming [4, 11] usually exploited for relaxing specific classes of stochastic programs and obtaining good bounds or approximate solutions. Nevertheless stochastic programming models are typically formulated as dynamic programs or MIP models. In both cases these bounds are not exploited for filtering decision variable domains as in our approach and they cannot be used for guiding the search.

4.7 Conclusions

We proposed a novel strategy to perform cost-based filtering for certain classes of stochastic constraint programs, under the assumptions that (i) the objective function is concave or convex in the stochastic variables, and (ii) the existence of a feasible solution is not affected by the distribution of the stochastic variables. This strategy is based on a known inequality borrowed from stochastic programming. We applied this technique to two combinatorial optimization problem involving uncertainty from the literature. Our results confirm that orders-of-magnitude improvements in terms of explored nodes and run times can be achieved. In the

future, we aim to apply cost-based filtering to multi-stage Stochastic COPs, define strategies to handle generic chance-constraints, which are currently ruled out by our assumptions, extend the approach to other valid inequalities such as Edmundson-Madansky [11] or to suitable inequalities for non-convex problems [57]. Finally, we plan to exploit the information provided by optimization-oriented global chance-constraints to define search strategies.

Chapter 5

Paper IV: Cost-based Filtering Techniques for Stochastic Inventory Control under Service Level Constraints

S. A. Tarim, B. Hnich, R. Rossi and S. Prestwich

Abstract

This paper considers a single product and a single stocking location production/inventory control problem given a non-stationary stochastic demand. Under a widely-used control policy for this type of inventory system, the objective is to find the optimal number of replenishments, their timings and their respective order-up-to-levels that meet customer demands to a required service level. We extend a known CP approach for this problem using three cost-based filtering methods. Our approach can solve to optimality instances of realistic size much more efficiently than previous approaches, often with no search effort at all.[†]

[†]This paper is an extended version of the work presented in [87].

5.1 Introduction

Inventory theory provides methods for managing and controlling inventories under different constraints and environments. An interesting class of production/inventory control problems is the one that considers the single-location, single-product case under non-stationary stochastic demand. Such a problem has been widely studied because of its key role in practice.

We consider the following inputs: a planning horizon of N periods and a demand d_t for each period $t \in \{1, \ldots, N\}$, which is a random variable with probability density function $g_t(d_t)$. In the following sections we will assume without loss of generality that these variables are normally distributed. We assume that the demand occurs instantaneously at the beginning of each time period. The demand we consider is non-stationary, that is it can vary from period to period, and we also assume that demands in different periods are independent. A fixed delivery cost a is considered for each order and also a linear holding cost h is considered for each unit of product carried in stock from one period to the next. Demands occurring when the system is out of stock are assumed to be back-ordered and satisfied as soon as the next replenishment order arrives. We assume that it is not possible to sell back excess items to the vendor at the end of a period. Our aim is to find a replenishment plan that minimizes the expected total cost, which is composed of ordering costs and holding costs, over the N-period planning horizon, satisfying the service level constraints. As a service level constraint we require that, with a probability of at least a given value α, at the end of each period the net inventory will be non-negative.

We decided to ignore in this model the linear production cost p, incurred for each unit produced. The logic behind this simplification of the problem is as follows. In the deterministic production planning problem, since all the demand has necessarily to be met, any optimal solution is independent of the given production cost. The production cost is therefore a constant of the problem. This is also true for the stochastic production planning problem under infinite horizon, provided that demands occurring when the system is out of stock are back-ordered and satisfied as soon as the next replenishment order arrives. Again the justification is that when time tends to infinity, under a demand back-ordering assumption, all the

realized demand will be necessarily satisfied and the production cost will become a constant of the problem. When the planning horizon is finite, as in our case, the production cost may have an impact on the structure of an optimal solution, as in an optimal solution we will tend to clear up stocks when we approach the end of the planning horizon. This may therefore affect the length of some replenishment cycles at the end of the planning horizon. In fact we may have a shorter final cycle in order to keep less buffer stocks at the very last period, especially if the production cost is high. On the other hand the proposed model has to be considered within the more general picture of inventory control. Typically a finite planning horizon assumption is made because forecasts cannot look too far ahead in time. This does not mean that production will stop at the end of the planning horizon: rather, a new optimization will often occur at that point, which considers new forecast information that has become available. This process is common in inventory control and it is known as a *rolling horizon* [81] approach. It is obvious that, under a rolling horizon approach and a demand back-ordering assumption, again in the long run we will tend to satisfy all the realized demand and the production cost will again become a constant of the problem as in the infinite horizon case. Moreover it should be noted that in this case considering a production cost p may even lead to suboptimal solutions, in fact we may schedule more replenishment cycles than strictly needed in order to keep unsold stocks low at the end of the given finite horizon. But since the production does not stop at the end of the finite horizon this will give no real cost benefit and will instead increase the total fixed delivery cost in the long run. For this reason we ignore such a cost component as Bookbinder and Tan do in their heuristic approach [15]. On the other hand extending the results in this paper to consider a production cost p is easy, and in Appendix 5.7.1 we will describe how this can be done. Different inventory control policies can be adopted for the described problem. A policy states the rules to decide when orders have to be placed and how to compute the replenishment lot-size for each order. For a discussion of inventory control policies see [81].

One of the possible policies that can be adopted is the replenishment cycle policy, (R, S).

Under the non-stationary demand assumption this policy takes the form (R^n, S^n) where R^n denotes the length of the nth replenishment cycle and S^n the order-up-

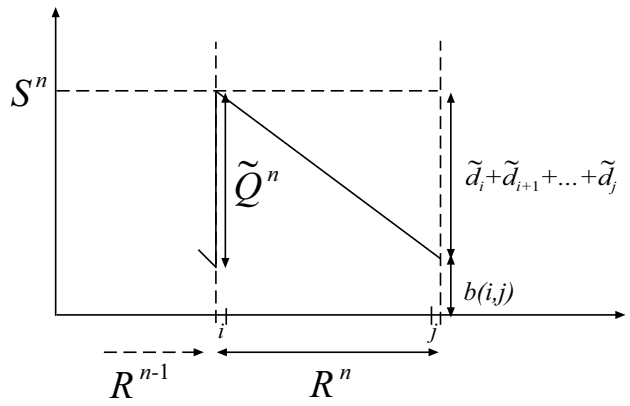


Figure 5.1: (R^n,S^n) policy. R^n denotes the set of periods covered by the nth replenishment cycle; S^n is the order-up-to-level for this cycle; $\tilde{Q}_n$ is the expected order quantity; $\tilde{d}_i + \tilde{d}_{i+1} + \ldots + \tilde{d}_j$ is the expected demand; $b(i, j)$ is the buffer stock required to guarantee the required service level α

to-level for replenishment (Fig. 5.1). In this policy a wait-and-see strategy is adopted, under which the actual order quantity Q_n for replenishment cycle n is determined only after the demand in former periods has been realized. The order quantity Q_n is computed as the amount of stock required to raise the closing inventory level of replenishment cycle $n-1$ up to level S^n. In order to provide a solution for our problem under the (R^n, S^n) policy we must populate both the sets R^n and S^n for $n = \{1, \ldots, N\}$.

There is a large literature on deterministic production planning. This problem has been mentioned by Garey and Johnson [37]. In [30] Florian et. al. gave an overview for the complexity of this problem. In particular they established NP-hardness for this problem under production cost (composed of a fixed cost and a variable unit cost), zero-holding cost and arbitrary production capacity constraint. They also extended this result by considering other possible cost functions and capacity constraints. Polynomial algorithms are discussed in the same paper for a few specific cases. Among these they cited Wagner and Whitin's [96] work, where the infinite capacity deterministic production planning problem is solved in polynomial time.

In contrast the respective stochastic formulation for this problem has been solved to optimality only recently, due to the complexity involved in the model-

ing of uncertainty and of the policy-of-response. Early works in this area adopted heuristic strategies such as those proposed by Silver [80], Askin [3] and Bookbinder & Tan [15]. Under some mild assumptions the first complete solution method for this problem was introduced by Tarim & Kingsman [89], who proposed a *deterministic equivalent* Mixed Integer Programming (MIP) formulation for computing (R^n, S^n) policy parameters. Empirical results showed that such a model is unable to solve large instances, but Tarim & Smith [92] introduced a more compact and efficient Constraint Programming (CP) formulation of the same problem that showed a significant computational improvement over the MIP formulation. A *stochastic constraint programming* [91] approach for computing (R^n, S^n) policy parameters is proposed in [75]. In this work the authors drop the mild assumptions originally introduced by Tarim & Kingsman and compute optimal (R^n, S^n) policy parameters. Of course there is a price to pay for dropping Tarim & Kingsman's assumptions, in fact this latter approach is less efficient than the one in [92].

This paper extends Tarim & Smith's work, which builds on Tarim & Kingsman's assumptions. We retain their model and we augment such a model with three *cost-based filtering* methods to enhance domain pruning. One of these techniques, based on a relaxation proposed by Tarim [86] and solved by means of dynamic programming, has been already presented in [87]. In this work we provide two additional cost-based filtering techniques and we extend the discussion on Tarim's relaxation and on the implementation of the respective cost-based filtering method.

Cost-based filtering is an elegant way of combining techniques from CP and Operations Research (OR) [28, 31]. OR-based optimization techniques are used to remove values from variable domains that cannot lead to better solutions. This type of domain filtering can be combined with the usual CP-based filtering methods and branching heuristics, yielding powerful hybrid search algorithms. Cost-based filtering is a novel technique that has been the subject of significant recent research, but to the best of our knowledge it has not previously been applied to stochastic inventory control. In the following sections we will show that it can bring a significant improvement when combined with the state-of-the-art CP model for stochastic inventory control. It should be noted that while the technique

based on Tarim's relaxation can easily be recognized as a classic *cost-based filtering* method, the two additional techniques here presented are not based on bounds obtained through a relaxation. Instead, as we will see, they exploit reasoning on the problem cost structure to prune values in the domains of decision variables that cannot lead to optimal solutions. Our experimental results show the efficiency obtained by the combined used of these three filtering techniques during the search for an optimal solution.

The paper is organized as follows. Section 5.2 describes the CP model and the pre-processing techniques introduced by Tarim & Smith. Section 5.3 firstly extends one of Tarim and Smith's pre-processing techniques to cost-based filtering method, allowing it to be applied at every search tree node. Secondly it proposes a general approach for applying any sound pre-processing technique at every search tree node in a cost-based filtering fashion. Section 5.4 describes a relaxation that can be efficiently solved by means of a shortest path algorithm, and produces tight lower bounds for the original problem which is used to perform further cost-based filtering. Section 5.5 evaluates our methods. Section 5.6 draws conclusions and discusses future extensions.

5.2 A CP model

In this section we review the CP formulation for the (R^n, S^n) policy proposed by Tarim & Smith [92]. First we provide some formal background related to stochastic programming.

Stochastic programming [11] is a well known modeling technique that deals with problems where uncertainty comes into play. Problems of optimization under uncertainty are characterized by the necessity of making decisions without knowing what their full effect will be. Such problems appear in many area of application and present many interesting conceptual and computational challenges. Stochastic programming needs to represent uncertain elements of the problem. Typically random variables are employed to model this uncertainty to which probability theory can be applied. For this purpose such uncertain elements must have a known probability distribution. The typical requirement in stochastic programs is to maintain certain constraints, called *chance constraints* [18], satisfied at a

prescribed level of probability. The objective is typically related to the minimization/maximization of some expectation on the problem costs. There are several different approaches to tackle stochastic programs. A first method dealing with stochastic parameters in stochastic programming is the so-called *expected value model* [11], which optimizes the expected objective function subject to some expected constraints. Another method, *chance-constrained programming*, was pioneered by Charnes and Cooper [18] as a means of handling uncertainty by specifying a confidence level at which it is desired that the stochastic constraint holds. Chance-constrained programming models can be converted into deterministic equivalents for some special cases, and then solved by some solution methods of deterministic mathematical programming. A typical example for this technique is given by the Newsvendor problem [81]. However it is almost impossible to do this for complex chance-constrained programming models. A third approach employs scenarios, which are particular representations of how the future might unfold. Each scenario is assigned a probability value, that is its likelihood. An appropriate probabilistic model or simulation is used to generate a batch of such scenarios. The challenge then, is how to make good use of these scenarios in coming up with an effective decision.

The stochastic programming formulation for the general multi-period production/inventory problem with stochastic demand can be expressed as finding the timing of the stock reviews and the size of the respective non-negative replenishment orders with the objective of minimizing the expected total cost $E\{TC\}$ over a finite planning horizon of N periods. The model is given below,

$$\begin{aligned} \min \; E\{TC\} = \int_{d_1}\int_{d_2}\dots\int_{d_N} \sum_{t=1}^{N} (a\delta_t + h\cdot \max(I_t,0)) \\ g_1(d_1)g_2(d_2)\dots g_N(d_N)\mathrm{d}(d_1)\mathrm{d}(d_2)\dots\mathrm{d}(d_N) \end{aligned} \tag{5.1}$$

subject to, for $t = 1 \ldots N$

$$\delta_t = \begin{cases} 1, & \text{if} \quad Q_t > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

$$I_t = I_0 + \sum_{i=1}^{t} (Q_i - d_i) \tag{5.3}$$

$$\Pr\{I_t \geq 0\} \geq \alpha \tag{5.4}$$

$$I_t \in \mathbb{Z}, \quad Q_t \geq 0, \quad \delta_t \in \{0, 1\}. \tag{5.5}$$

Each decision variable I_t represents the inventory level at the end of period t. The binary decision variables δ_t state whether a replenishment is fixed for period t ($\delta_t = 1$) or not ($\delta_t = 0$). If an order is placed in period t, constraint (5.2), decision variable Q_t denotes the size of the respective non-negative replenishment order. Chance constraint (5.4) enforces the required service level, that is the probability α that the net inventory will not be negative at the end of each time period. The objective function (5.1) minimizes the expected total cost over the given planning horizon.

In [89] the authors assume that negative orders are not allowed, so that if the actual stock exceeds the order-up-to-level for that period, this excess stock is carried forward and not returned to the supply source. However, such occurrences are regarded as rare events and accordingly the cost of carrying the excess stock and its effect on the service level of subsequent periods is ignored. Under these assumptions the chance-constrained problem can be expressed by means of a *deterministic equivalent* model where buffer stocks for each possible replenishment cycle are computed independently.

We now recall some basic notions about *constraint programming*. A *Constraint Satisfaction Problem* (CSP) [1, 17] is a triple $\langle V, C, D \rangle$, where V is a set of decision variables each with a discrete domain of values $D(V_k)$, and C is a set of constraints stating allowed combinations of values for subsets of variables in V. Finding a solution to a CSP means assigning values to variables from the domains without violating any constraint in C. We may also be interested in finding a feasible solution that minimizes (maximizes) the value of a given objective function over a subset of the variables. Constraint solvers typically explore par-

tial assignments enforcing a local consistency property using either specialized or general purpose propagation algorithms. Such propagation algorithms in general exploit some structure of the problem to prune decision variable domains in more efficient ways.

The following CP formulation of the *deterministic equivalent* model for the (R^n, S^n) policy is proposed in [92]:

$$\min \; E\{TC\} = \sum_{t=1}^{N} \left(a\delta_t + h\tilde{I}_t \right) \tag{5.6}$$

subject to, for $t = 1 \ldots N$

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} \geq 0 \tag{5.7}$$

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} > 0 \Rightarrow \delta_t = 1 \tag{5.8}$$

$$\tilde{I}_t \geq b \left(\max_{j \in \{1,\ldots,t\}} j \cdot \delta_j, t \right) \tag{5.9}$$

$$\tilde{I}_t \in \mathbb{Z}^+ \cup \{0\}, \quad \delta_t \in \{0, 1\}, \tag{5.10}$$

where $b(i, j)$ is defined by

$$b(i, j) = G^{-1}_{d_i + d_{i+1} + \ldots + d_j}(\alpha) - \sum_{k=i}^{j} \tilde{d}_k.$$

Constraint (5.9), originally proposed by Tarim and Smith, can be implemented by means of the following set of constraints, for $t = 1 \ldots N$

$$Y_t \geq j \cdot \delta_j \quad j = 1, \ldots, t \tag{5.11}$$

$$element\,(Y_t, b(\cdot, t), H_t) \tag{5.12}$$

$$\tilde{I}_t \geq H_t \tag{5.13}$$

$$\tilde{I}_t, H_t \in \mathbb{Z}^+ \cup \{0\}, \quad \delta_t \in \{0, 1\}, \quad Y_t \in \{1, \ldots, N\}. \tag{5.14}$$

The $element(X, list[], Y)$ constraint [45] enforces a relation such that variable Y represents the value of element at position X in the given list. $G_{d_i + d_{i+1} + \ldots + d_j}$

is the cumulative probability distribution function of $d_i + d_{i+1} + \ldots + d_j$. It is assumed that G is strictly increasing, hence G^{-1} is uniquely defined.

Each decision variable $\tilde{I}_t$ represents the expected inventory level at the end of period t. Each $\tilde{d}_t$ represents the expected value of the demand in a given period t according to its probability density function $g_t(d_t)$. The binary decision variables δ_t state whether a replenishment is fixed for period t ($\delta_t = 1$) or not ($\delta_t = 0$). The objective function (5.6) minimizes the expected total cost over the given planning horizon. The two terms that contribute to the expected total cost are ordering costs and inventory holding costs. Constraint (5.7) enforces a no-buy-back condition, which means that received goods cannot be returned to the supplier. As a consequence of this the expected inventory level at the end of period t must be no less than the expected inventory level at the end of period $t - 1$ minus the expected demand in period t. Constraint (5.8) expresses the replenishment condition. We have a replenishment if the expected inventory level at the end of period t is greater than the expected inventory level at the end of period $t - 1$ minus the expected demand in period t. This means that we received some extra goods as a consequence of an order. Constraints (5.9) enforce the required service level α. This is done by specifying the minimum buffer stock required for each period t in order to assure that, at the end of each and every time period, the probability that the net inventory will not be negative is at least α. These buffer stocks, which are stored in matrix $b(\cdot, \cdot)$, are pre-computed following the approach suggested in [89]. In this approach the authors transformed a chance-constrained model, that is a model where constraints on some random variables have to be maintained at prescribed levels of probability, in a completely deterministic one. For further details about chance-constrained programming see [18].

5.2.1 Domain pre-processing

In [92] the authors showed that a CP formulation for computing optimal (R^n, S^n) policies provides a more natural way of modeling the problem. In contrast to the equivalent MIP formulation the CP model requires fewer constraints and provides a neater formulation. However, the CP model has two major drawbacks. Firstly, in order to improve the search process and quickly prove optimality, tight bounds on

the objective function are needed. Secondly, even when it is possible to compute *a priori* the maximum values that such variables can be assigned to, these values (and therefore the domain sizes of the $\tilde{I}_t$ variables) are large. The domain size value is equal to the amount of stock required to satisfy subsequent demands till the end of the planning horizon, meeting the required service level when only a single replenishment is scheduled at the beginning of the planning horizon.

To address the domain size issue, Tarim & Smith proposed two pre-processing methods in order to reduce the size of the domains before starting the search process, by exploiting properties of the given model and of the (R^n, S^n) policy. Method I computes a cost-based upper bound for the length of each possible replenishment cycle $T(i, j)$, starting in period i, for all $i, j \in \{1, \ldots, N\}$, $i \leq j$. Note that $T(i, j)$ denotes the time span between two consecutive replenishment periods i and $j + 1$. Method I therefore identifies sub-optimal replenishment cycle lengths allowing a proactive off-line pruning, which eliminates all the expected inventory levels that refer to longer sub-optimal replenishment cycles. Method II employs a dynamic programming approach, by considering each period in an iterative fashion and by taking into account in each step two possible courses of action: either an order with an expected size greater than zero is placed, or no order (equivalently an order with a null expected size) is placed in the considered period within our planning horizon. The effects of these possible actions in each step are reflected in the decision variable domains by removing values that are not produced by any course of action.

5.3 From pre-processing to cost-based filtering

In the previous section we described a CP formulation for the (R^n,S^n) policy. In [92] the authors discussed the advantages of such a formulation when it is compared to the MIP formulation proposed in [89]. CP not only performs faster than MIP and provides a neater formulation, it also allows us to build dedicated filtering algorithms for pruning infeasible and/or suboptimal values for the domains of decision variables during the search.

In Section 5.3.1 we extend the first of the two pre-processing methods proposed in [92] in order to exploit partial assignments of decision variables in the

model to prune suboptimal values from the domains of the remaining decision variables still unassigned at any point of the search process.

In Section 5.3.2, we describe a generic approach to applying pre-processing techniques not only in a proactive way, before the search process starts, but also during the search, by exploiting partial information which derives from the current decision variable assignments. We emphasize that this approach may be used in conjunction with any sound pre-processing method developed for our inventory/production problem and it is not limited to the two pre-processing methods proposed in [92].

A running example is given to show that the two methods proposed are incomparable in term of domain reduction achieved.

5.3.1 Tighter upper bounds for optimal replenishment cycle lengths

We now present a filtering method that is a natural extension of pre-processing method I in [92]. This method prunes variable domains, when a partial solution is given, by enforcing tighter upper bounds for optimal replenishment cycle lengths than those proposed by Tarim and Smith. When no partial solution is provided this filtering method realizes the same domain reduction performed by the respective pre-processing method.

Firstly let $R(i,j) = b(i,j) + \sum_{t=i}^{j} \tilde{d}_t$ be the required minimum opening inventory level in period i, $i \in \{1,\ldots,N\}$, to meet demand until period $j+1$. The cycle cost $c(i,j)$, when a variable holding cost h_t ($t \in \{1,\ldots,N\}$) is considered, can be expressed as

$$c(i,j) = a + \sum_{t=i}^{j} h_t b(i,j) + \sum_{t=i}^{j-1} h_t \sum_{k=t+1}^{j} \tilde{d}_k. \tag{5.15}$$

The cost (5.15) of a replenishment cycle is the sum of two components. A fixed ordering cost a, that is charged at the beginning of the cycle when an order is placed, and a variable holding cost h_t charged at the end of each time period within the replenishment cycle and proportional to the amount of stocks held in

inventory. In [92], for each period $i \in \{1, \ldots, N\}$ over the planning horizon N, an upper bound for the length of an optimal replenishment cycle $T(i,p)^*$ that starts in such a period is proposed. The authors compute *a priori* this bound for every period i and derive from it a superset of all candidate opening-inventory-levels for any period in the planning horizon. Let us refer to this bound as B (Fig. 5.2 - a), and let $j = i + B$. Then the last period p of an optimal replenishment cycle $T(i,p)^*$ satisfies $i \leq p \leq j$. $j = i + B$ can be computed as the minimum

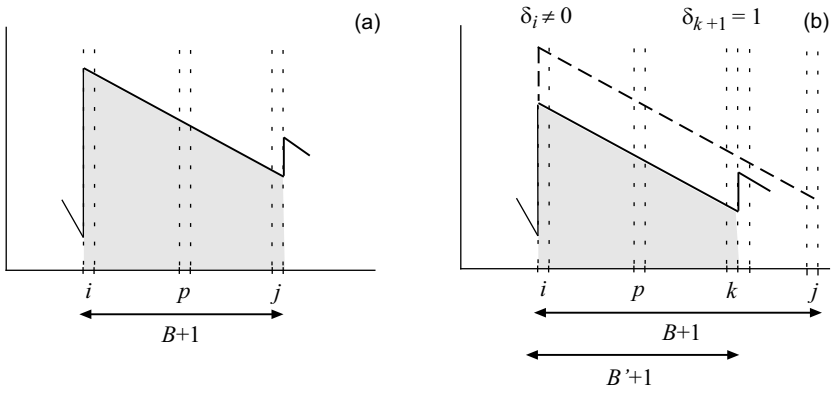


Figure 5.2: Bound tightening when a partial solution is given: (a) since it is not optimal to cover more than $B + 1$ periods with a single replenishment in i, the optimal policy lies in the gray area; (b) the bound B can be tightened to B' when an order is scheduled in period $k + 1$, $i \leq k < j$

j satisfying the following conditions described in [92], which formally identify bound B

$$c(i,k) + c(k+1,j) > c(i,j) \vee b(i,k) > R(k+1,j) \tag{5.16}$$

for all $k \in \{i, \ldots, j-1\}$, and

$$c(i,k) + c(k+1,j+1) \leq c(i,j+1) \wedge b(i,k) \leq R(k+1,j+1) \tag{5.17}$$

for some $k \in \{i, \ldots, j\}$, given that $\forall p \in \{j+2, \ldots, N\}$ such a k satisfies

$$-\sum_{t=j+2}^{p}(k+1-i)\tilde{d}_t + (p-k)b(k+1,p) - (j-k+1)b(k+1,j+1) \leq (p-i+1)b(i,p) - (j-i+2)b(i,j+1). \tag{5.18}$$

A proof for these conditions is given in Appendix 5.7.2.

When a partial solution S is given, it is possible to tighten the bound B by using the following observations:

- if δ_i is assigned to 0 then no replenishment cycle starts in period i.
- if δ_i is not assigned to 0 and $\exists k \in \{i, \ldots, i+B-1\}$ such that $\delta_{k+1} = 1$, then B can be tightened to the smallest $k - i$ value B' (Fig. 5.2 - b)

In order to compute the tighter bound B' for a given period $i \in \{1, \ldots, N\}$ when a partial solution S is given we introduce the following Lemma.

Lemma 5.3.1. *If there exists some $k \in S$ such that $\delta_{k+1} = 1$ and $i \leq k < j$, then B can be tightened to $B' = j' - i$ where*

$$j' = \min\left(\{k|\ \delta_{k+1} = 1, k \in \{i, \ldots, j-1\}\} \quad \bigcup \quad \{j\}\right).$$

Proof. Trivially the replenishment scheduled in period $k+1$ rules out the chance of covering periods $i, \ldots, j$ where $j > k$ with a single cycle. □

By means of the described tighter bound B' we can now obtain smaller supersets of all candidate opening-inventory-levels than those described in [92]. For convenience in what follows we will refer to the expected closing-inventory-levels, that is opening-inventory-level minus expected demand in the period considered.

A first reduction in the size of the super-sets is due to the fact that if δ_i is assigned to zero, no replenishment cycle starts in period i. Therefore no value that is a candidate expected closing-inventory-level for any replenishment cycle starting in period i is feasible with respect to the given partial solution. Otherwise candidate values can be computed as described in the following:

Lemma 5.3.2. *When δ_i is not assigned to 0, a sufficient but not necessary condition that identifies candidate expected closing-inventory-level values in $Dom(\tilde{I}_m)$, $m \in T(i, j')$ for a replenishment cycle starting in period i is defined as follows (see Fig. 5.3):*

$$Dom(\tilde{I}_m) \supseteq \left\{ \tau \;\middle|\; \tau = R(i,l) - \sum_{t=i}^{m} \tilde{d}_t,\ l \in \{m, \ldots, j'\} \right\}. \tag{5.19}$$

Proof. As shown in [92], equation (5.19) considers in $Dom(\tilde{I}_m)$ for each $m \in T(i, j')$ every value that is feasible if there is a replenishment cycle starting in period i. In fact if p denotes the final period of the optimum length replenishment cycle for period i, $\delta_k = 0$, $k = \{i+1, \ldots, p\}$, the optimum expected closing inventory level for period m, where $i \leq m \leq p$, is $R(i,p) - \sum_{t=i}^{m} \tilde{d}_t$. The domain of possible values is therefore obtained by letting p range from m to j. Tightening j to j' is correct because, when a partial solution is given, this ignores values related to every infeasible replenishment cycles $T(i,r)$, where $j' < r \leq j$ and $\delta_{j'+1} = 1$, if any exists. □

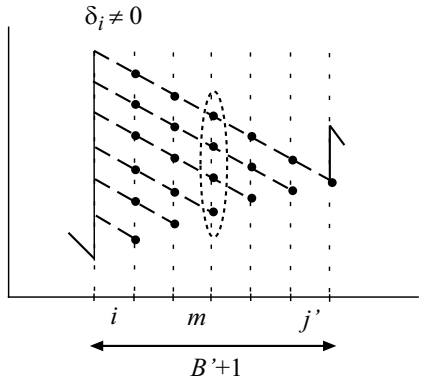


Figure 5.3: Subset of candidate optimal expected closing-inventory-levels for period m, $m \in \{i, \ldots, j'\}$. These values can be computed as stated in Lemma 5.3.2. The whole set of candidate levels shown in the picture may be computed by ranging m from i to j'

The former condition is only sufficient because there may exist other candidate values that should be in $Dom(\tilde{I}_m)$ as we did not take into account *negative order quantity* scenarios. Such situations arise when for some $m \in T(i, j')$, $c(i,m) +$

$c(m+1, j') \leq c(i, j')$ and $b(i, m) > R(m+1, j')$ (Fig. 5.4 - a). In this case, since

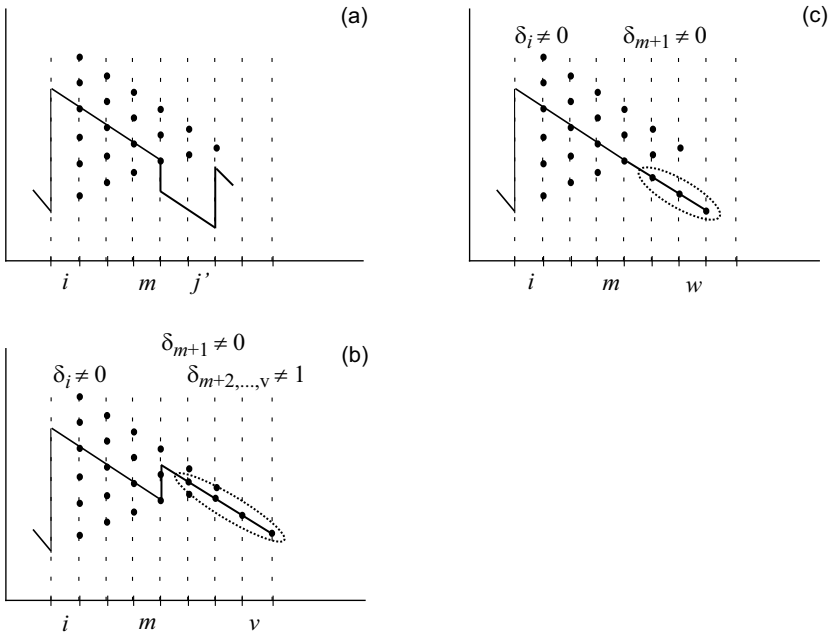


Figure 5.4: (a) Negative order quantity scenario. Additional values, computed by Lemma 5.3.3, to be considered in the subset of candidate optimal expected closing-inventory-levels for each period p when (b) an order with expected size greater than zero is scheduled in period $m+1$, $p \in \{m+1, \ldots, h'\}$, (c) an order with expected size zero is scheduled in period $m+1$, $p \in \{m+1, \ldots, w\}$. In both cases $\delta_{m+1} \neq 0$ since it must be possible to schedule an order in period $m+1$

the replenishment policy expects a negative order and is infeasible, an optimal policy can be either the one that schedules a new order in period $m+1$ with an expected lot-size greater than zero (Fig. 5.4 - b) or an expected lot-size of zero (Fig. 5.4 - c). Lemma 5.3.3 and 5.3.4 characterize which additional values have to be considered when a negative order quantity scenario arises.

Lemma 5.3.3. *If $\delta_{m+1} = 0$, Eq. (5.19) is a necessary and sufficient condition that identifies candidate expected closing-inventory-level values in $Dom(\tilde{I}_m)$, $m \in T(i, j')$ for a replenishment cycle starting in period i.*

Proof. In [92] it is stated that, if i is a replenishment period and we want to cover subsequent periods up to m, in a feasible policy a replenishment should then be scheduled in $m+1$. Since $\delta_{m+1} = 0$, it is not feasible to cover periods from i to

m with a single order in i because to do so we would need an additional order in period $m+1$ that is ruled out by the partial assignment. □

Lemma 5.3.4. *If δ_{m+1} is not assigned to zero, every further candidate expected closing-inventory-level value for a replenishment cycle starting in period i can be identified by considering two possible courses of action:*

- *a new order is scheduled for period $m+1$ and its expected size is greater than zero, (Fig. 5.4 - b). In this case, if $\delta_k \neq 1$ for $k = \{m+2, \ldots, v\}$, we also consider the following candidate expected closing-inventory levels*

$$Dom(\tilde{I}_n) \supseteq \left\{ \tau \;\middle|\; \tau = R_{(m+1,v)} - \sum_{t=m+1}^{n} \tilde{d}_t \right\}, \tag{5.20}$$

  for $n = \{m+1, \ldots, v\}$, where $v = \min\left\{ l \,\middle|\, b(m+1,l) + \sum_{t=m+1}^{l} \tilde{d}_t \geq b(i,m) \right\}$.

- *a new order is scheduled for period $m+1$ and its expected size is zero, (Fig. 5.4 - c). In this case we also consider the following candidate expected closing-inventory levels*

$$Dom(\tilde{I}_n) \supseteq \left\{ \tau \;\middle|\; \tau = b_{(i,m)} - \sum_{t=m+1}^{n} \tilde{d}_t \right\}, \tag{5.21}$$

  for $n \in \{m+1, \ldots, w\}$, where

$$w = \max\left\{ l \,\middle|\, \exists q \in \{m+1, \ldots, l\}, b(q,l) + \sum_{t=m+1}^{l} \tilde{d}_t \leq b(i,m) \right\}.$$

Proof. As shown in [92], equation (5.20) adds to $Dom(\tilde{I}_n)$ every further feasible values by considering the option of placing an order whose expected lot-size is bigger than zero. In fact if we assume that the high levels of opening inventory carried from period m satisfy the service-level constraint for the following $v-1$ consecutive periods, then the remaining inventory is not enough to satisfy this constraint for period v. To comply with the service level constraint in period v, the order quantity must be at least $b(m+1,v) + \sum_{t=m+1}^{v} \tilde{d}_t - b(i,m)$. Hence

this replenishment covers the periods until the end of v, where $v = \min\{l | b(m+1, l) + \sum_{t=m+1}^{l} \tilde{d}_t \geq b(i, m)\}$. If an order has been scheduled for a period $t \in \{m+2, \ldots, v\}$, then by definition the remaining inventory at the end of period m is enough to satisfy demands in periods $\{m+1, \ldots, t\}$, therefore the optimal expected order quantity for period $m+1$ is zero.

Equation (5.21) adds to $Dom(\tilde{I}_n)$ every further feasible values by considering the option of placing an order whose expected lot-size is zero. In this case, since the replenishment expects a zero order quantity, the excess stock may affect subsequent periods regardless of the orders placed. Therefore we look forward in the planning horizon up to the point where no following replenishment cycle may be affected by the excess stock carried on from the current one. Hence, the farthest period that may be affected is $w = \max\{l | \exists q \in \{m+1, \ldots, l\}, b(q, l) + \sum_{t=m+1}^{l} \tilde{d}_t \leq b(i, m)\}$. □

Theorem 5.3.1. *When a partial solution is given, by ranging i from 1 to N, equations (5.19, 5.20, 5.21) identify the feasible subset of values within the current $Dom(\tilde{I}_k)$, for $k \in \{1, \ldots, N\}$.*

Proof. Directly follows from Lemmas 5.3.1, 5.3.2, 5.3.3 and 5.3.4. □

Example 5.3.1. We now present a running example where the planning horizon is $N = 24$ periods and the initial stock level is equal to zero. The demand is normally distributed in each period $t \in \{1, ..., N\}$ with a constant coefficient of variation $\sigma_t / \tilde{d}_t = 1/3$, where σ_t is the standard deviation of the demand in period t. The demand forecasts (mean value for each period) are listed in Table 5.1. The other parameters for the problem are: $a = 200$, $h = 1$, $\alpha = 0.95$. The optimal solution for the CP model when former inputs are considered is shown in Table 5.2. The (R^n, S^n) policy parameters, that is replenishment cycle lengths and order-up-to-levels, for this instance can be easily computed from the solution of the CP model. We applied the described filtering method without considering a given partial solution, the domain reduction achieved is therefore equivalent to the one performed by pre-processing method I introduced in [92]. This way we computed the reduced domains $Dom(I_t)$ for the decision variables I_t, $t \in \{1, ..., N\}$. These reduced domains are shown in Table 5.3. We now consider the partial solution shown in Table 5.4. Table 5.5 shows the reduced domains obtained when we en-

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\tilde{d}_i$	73	0	128	116	92	180	28	164	28	161	37	57	181	62
i	15	16	17	18	19	20	21	22	23	24				
$\tilde{d}_i$	34	161	2	10	40	192	17	190	163	32				

Table 5.1: Demand forecasts

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
δ_i	1	0	1	1	0	1	0	1	0	1	1	0	1	1
$\tilde{I}_i$	40	40	70	173	81	128	100	119	91	88	94	37	99	73
i	15	16	17	18	19	20	21	22	23	24				
δ_i	0	1	1	0	0	1	0	1	1	0				
$\tilde{I}_i$	39	88	86	76	36	123	106	104	123	91				

Table 5.2: Optimal solution

i	$Dom(\tilde{I}_i)$	i	$Dom(\tilde{I}_i)$
1	$\{\underline{40}\}$	13	$\{\underline{99}, 167\}$
2	$\{0, \underline{40}, 198\}$	14	$\{34, 37, \underline{73}, 105\}$
3	$\{\underline{70}, 211\}$	15	$\{19, \underline{39}\}$
4	$\{64, 95, \underline{173}\}$	16	$\{\underline{88}, 90, 100, 143\}$
5	$\{50, \underline{81}\}$	17	$\{1, 16, 73, \underline{86}, 88, 98, 141, 350\}$
6	$\{99, \underline{128}\}$	18	$\{5, 6, 63, \underline{76}, 78, 88, 131, 340\}$
7	$\{15, 71, \underline{100}\}$	19	$\{22, 23, \underline{36}, 38, 91, 300\}$
8	$\{90, \underline{119}\}$	20	$\{105, 108, \underline{123}\}$
9	$\{15, 62, \underline{91}\}$	21	$\{9, 88, \underline{106}\}$
10	$\{\underline{88}, 128\}$	22	$\{\underline{104}\}$
11	$\{20, 51, 91, \underline{94}\}$	23	$\{89, \underline{123}\}$
12	$\{31, \underline{37}\}$	24	$\{18, 57, \underline{91}\}$

Table 5.3: Reduced domains after applying our filtering method when no partial solution is given. The reduction achieved is equivalent to the one provided by pre-processing method I in [92]. Underlined figures are closing inventory levels of the optimal policy

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
δ_i	1	0	1	–	0	1	0	1	0	–	–	0	1	–
i	15	16	17	18	19	20	21	22	23	24				
δ_i	0	1	1	0	0	1	0	1	–	0				

Table 5.4: Partial solution. A "–" means that the variable has not been assigned yet

force tighter upper bounds for optimal replenishment cycle lengths considering the partial solution in Table 5.4. From Theorem 5.3.1 it directly follows that the filtering is performed by removing from decision variables domains (Table 5.3) values that do not appear in Table 5.5, which contains the computed reduced do-

mains with respect to the partial solution given.

We shall now see in details how feasible expected closing-inventory-levels in the reduced domains (Table 5.5) are computed for the first 5 periods. In the given partial solution we place an order in period 1 but not in period 2. An order is placed in period 3 therefore a replenishment cycle over periods $\{1, 2\}$ is uniquely defined. Bound B' for period 1 is 2 periods. The demand in the first period is 73 while in the second is 0. The buffer stock required at the end of period 1 is $70 \cdot 1.645 \cdot 0.3 \simeq 40$. By iterating Lemma 5.3.2 over periods $\{1, 2\}$ we obtain an expected closing-inventory-level of 40 for period 1 and again of 40 for period 2. Negative order quantity scenarios do not arise since $\delta_2 = 0$. We do not iterate Lemma 5.3.2 for period 2, since $\delta_2 = 0$ and no replenishment cycle may start in this period. In period 3 a replenishment is scheduled. The replenishment decision in period 4 is still unassigned while in period 5 no replenishment is scheduled. We apply Lemma 5.3.2 to period 3. The bound B' is 2 periods. Therefore either we may cover only the current period with a replenishment, which yields a closing inventory level of 70, or we may cover both the periods with a single replenishment, in which case the required expected closing-inventory-level is 211 in period 3 and 95 in period 4. Negative order quantity scenarios do not arise. In period 4 the bound B' is again 2. Therefore we may cover only one period with an expected closing-inventory-level of 64, or we may cover two periods by keeping respectively an expected closing-inventory-level of 173 at the end of period 4 and of 81 at the end of period 5. Negative order quantity scenario again do not arise. δ_5 is assigned to 0 therefore no replenishment cycle starts in this period.

We now consider a set of periods where negative order quantity scenarios arise. We refer to periods $\{10, 11, 12\}$. In period 10, B' is 2 periods. Therefore the two candidate expected closing inventory levels computed by Lemma 5.3.2 are $\{88, 128\}$. 88 is the expected closing-inventory-level required if only one period is covered by the replenishment scheduled in period 10, 128 is the level required to cover period 10 and 11 with a single replenishment. In this case the respective expected closing-inventory-level at the end of period 11 is 91. If an order is placed in period 10 and also in period 11 the overall cost is higher than that incurred by covering both the periods with a single replenishment. On the other hand the order-up-to-level for period 11 in this case is lower than the expected closing-

i	$Dom(I_i)$	i	$Dom(I_i)$
1	$\{\underline{40}\}$	13	$\{\underline{99}, 167\}$
2	$\{\underline{40}\}$	14	$\{34, 37, \underline{73}, 105\}$
3	$\{\underline{70}, 211\}$	15	$\{\underline{39}\}$
4	$\{64, 95, \underline{173}\}$	16	$\{\underline{88}\}$
5	$\{\underline{81}\}$	17	$\{1, 16, 73, \underline{86}\}$
6	$\{99, \underline{128}\}$	18	$\{6, 63, \underline{76}\}$
7	$\{\underline{100}\}$	19	$\{23, \underline{36}\}$
8	$\{90, \underline{119}\}$	20	$\{105, \underline{123}\}$
9	$\{\underline{91}\}$	21	$\{\underline{106}\}$
10	$\{\underline{88}, 128\}$	22	$\{\underline{104}\}$
11	$\{20, 51, 91, \underline{94}\}$	23	$\{89, \underline{123}\}$
12	$\{\underline{37}\}$	24	$\{\underline{91}\}$

Table 5.5: Enforcing tighter upper bounds for optimal replenishment cycle lengths - Partial solution in Table 5.4, underlined figures are closing inventory levels of the optimal policy

inventory-level in period 10. This generates a negative order quantity scenario. As stated in Lemma 5.3.4, either we cover period 11 only by scheduling an order with expected size zero. In this case the candidate level $51 = 88 - 37$ must be considered for period 11. Otherwise we try to cover more periods with the candidate level 94. By doing so we will cover subsequent periods till 12, therefore we add the candidate level $37 = 94 - 57$ to period 12. The other value in the table for period 11 is 20 that refers instead to the case in which we order in this period and we cover only 1 period with the order. This value is computed by applying Lemma 5.3.2 to this period. Since $\delta_{12} = 0$ no replenishment cycle may start in this period. $\diamond$

5.3.2 Merging adjacent non-replenishment periods

One of the limits of the domain reduction methods proposed in [92] is that they can only be applied before the search process starts. Therefore they do not take into account information regarding partial assignments for decision variables that may become available during the search process. In this section we aim to overcome this limitation with a general approach that may be applied to any pre-processing method.

We consider a given partial solution in which some decision variables δ_i are set to zero. The key idea is to transform the original problem instance into a smaller

one by merging adjacent non-replenishment periods into a single new period with new expected demand and variance values. Since the demand in each period is assumed to be independent from the previous and the following demands, these new characteristics for the demand distribution in the new merged time span can be easily computed by exploiting properties of the chosen probability distribution. Once we have the smaller instance fully defined, we can apply any sound pre-processing methods, for instance one of those presented in [92], and then we can reflect the pruning achieved in the smaller instance back onto the original one. It should be noted that the following reasoning can be applied to any reduction method for the presented CP model, and it is not limited to those presented in [92]. We propose a three-step procedure to apply any pre-processing method not only at the root node, but at every node of the search tree.

Step 1 By considering a partial solution S for the original problem instance $\mathcal{P}$, we construct a reduced problem instance $\mathcal{R}$. $\mathcal{R}$ will be described by a list of $M \leq N$ expected demand values and standard deviations and it will be built as follows. If $\delta_k = 0$ for all $k \in \{i+1, \ldots, j\}$ and $\delta_i = 1$ or δ_i is unassigned, then instead of periods $\{i, \ldots, j\}$ we introduce a new period k^* that represents such a span with an expected demand of

$$\tilde{d}_{k^*} = \sum_{t=i}^{j} \tilde{d}_t$$

and a standard deviation of

$$\sigma_{k^*} = \sqrt{\sum_{t=i}^{j} \sigma_t^2}.$$

These two expressions are well known properties of the normal distribution. The holding cost for period k^* can be expressed as $h \cdot (j-i+1) I_{k^*} + \sum_{l=i+1}^{j} (l-i)\tilde{d}_l$, and since the second term is constant the new holding cost coefficient will be $h_{k^*} = h \cdot (j-i+1)$. For any other period in $\mathcal{P}$ we introduce a duplicate period in $\mathcal{R}$ with the same expected demand, variance and holding cost. To avoid confusion, we will refer to the decision variables denoting the closing inventory level at period i in problem $\mathcal{R}$ as $\tilde{I'}_i$, to the binary variables as δ'_i, for all $i \in \{1, \ldots, M\}$ and to

the demands as $\tilde{d'_i}$, for all $i \in \{1, \ldots, M\}$.

Step 2 In this step we apply a sound pre-processing method to the reduced problem instance $\mathcal{R}$ defined in the previous step.

Step 3 In this step we reflect the pruning done in the reduced instance back to the original instance. For each period $p \in \{1, \ldots, M\}$ of $\mathcal{R}$ that is the result of merging adjacent periods $\{i, \ldots, j\}$, $i < j$ of $\mathcal{P}$, we can update the domains of $\tilde{I}_t$ for all $i \leq t \leq j$ by enforcing the following constraints:

$$\tilde{I}_t = \begin{cases} \tilde{I'_p} & \text{if } t = j, \\ \tilde{I'_p} + \tilde{d}_j + \tilde{d}_{j-1} + \ldots + \tilde{d}_{t-1} & \text{if } i \leq t < j. \end{cases} \tag{5.22}$$

For any other period $p \in \mathcal{R}$ that does not represent merged periods and its corresponding period t in $\mathcal{P}$, we enforce that

$$\tilde{I}_t = \tilde{I'_p}. \tag{5.23}$$

These three steps compose the core of our algorithm. The following Theorem shows that such a filtering algorithm is sound.

Theorem 5.3.2. *We are given a problem instance $\mathcal{P}$ and a partial solution S for it, where $\exists \delta_i$, $i \in \{1, \ldots, N\}$ such that $\delta_i = 0$. By applying a sound pre-processing method (Step 2) to the reduced problem instance $\mathcal{R}$, obtained as described in Step 1, and by computing feasible values for decision variables $\tilde{I}_t$ in the original problem $\mathcal{P}$, as stated in Step 3, no value that is part of any optimal solution S^* with respect to the given partial assignments in S is pruned in the domain of $\tilde{I}_t$, $t \in \{1, \ldots, N\}$.*

Proof. We will now show that, under the given partial solution S, the reduced problem instance $\mathcal{R}$ is equivalent to the original problem $\mathcal{P}$ and that the reduction in the number of decision variables and constraints is a direct consequence of the linear dependencies induced by the current partial assignment for δ_t variables. This will establish the fact that any sound pre-processing method applied to $\mathcal{R}$ will

produce a sound domain reduction in $\mathcal{P}$ when reflected by means of the proposed mapping that is built on these linear dependencies.

Let us consider the model above for our problem $\mathcal{P}$ that is defined by Eqs. 5.6, 5.7, 5.8, 5.9 and 5.10.

Consider $\mathcal{P}$ and a partial solution where $\exists k \in \{1, ..., N\}$ s.t. δ_k is set to 0. Let us consider the implications of this assignment in our model $\mathcal{P}$. This assignment affects the *inventory conservation constraints* 5.7 and obviously the *replenishment decisions* 5.8, the *constraints that enforce buffer stocks* 5.9 and the *objective function* 5.6.

Effects on the replenishment decision and on the inventory conservation constraints. Since $\delta_k = 0$, constraint 5.7 for $t = k$ can be tightened because of Eq. 5.8 as follows:

$$\tilde{I}_k + \tilde{d}_k - \tilde{I}_{k-1} = 0, \tag{5.24}$$

then, by using $\tilde{I}_{k-1} + \tilde{d}_{k-1} - \tilde{I}_{k-2} \geq 0$ (that is constraint 5.7 for $t = k - 1$) and Eq. 5.24, we have

$$\tilde{I}_k + \tilde{d}_k + \tilde{d}_{k-1} - \tilde{I}_{k-2} \geq 0. \tag{5.25}$$

Notice that constraint 5.8 for $t = k$ is now redundant, since we assume that $\delta_k = 0$. Furthermore by following a reasoning similar to the one used to derive Eq. 5.25, Eq. 5.8 for $t = k - 1$ can be replaced by the following constraint

$$\tilde{I}_k + \tilde{d}_k + \tilde{d}_{k-1} - \tilde{I}_{k-2} > 0 \rightarrow \delta_{k-1} = 1. \tag{5.26}$$

Effects on the constraints that enforce buffer stocks. Let us consider now the implications of constraint 5.24 on the buffer stock levels. When $t = k - 1$ in constraint 5.9 we can write

$$\tilde{I}_k + \tilde{d}_k \geq b \left(\max_{j \in \{1,...,k-1\}} j \cdot \delta_j, k - 1 \right). \tag{5.27}$$

Also notice that for $t = k$

$$\tilde{I}_k \geq b \left(\max_{j \in \{1,...,k\}} j \cdot \delta_j, k \right) \tag{5.28}$$

and since $\delta_k = 0$, Eq. 5.28 can be rewritten as

$$\tilde{I}_k \geq b\left(\max_{j\in\{1,\ldots,k-1\}} j\cdot\delta_j, k\right). \tag{5.29}$$

Since the buffer stock level $b(i,j)$ is an increasing function of the number of periods as shown in [92], it is easy to see that

$$\tilde{I}_k \geq b\left(\max_{j\in\{1,\ldots,k-1\}} j\cdot\delta_j, k\right) \geq b\left(\max_{j\in\{1,\ldots,k-1\}} j\cdot\delta_j, k-1\right), \tag{5.30}$$

it follows that Eq. 5.27 (that is constraint 5.9 for $t = k-1$) becomes redundant.

Effects on the objective function. We now consider the implications of constraint 5.24 on the objective function. Since $\delta_k = 0$ the fixed ordering cost component for period k is zero. By applying constraint 5.24 we obtain the following new objective function

$$\min \;\; E\{TC\} = \sum_{t=1,t\neq k}^{N} a\delta_t + \sum_{t=1,t\neq k-1}^{N} h\tilde{I}_t + h(\tilde{I}_k + \tilde{d}_k). \tag{5.31}$$

We can see that we no longer have a holding cost component for period $k-1$, while the holding cost for period k is now doubled, since we can ignore the constant term $h\cdot\tilde{d}_k$.

Every implication of Eq. 5.24 in the whole model has been considered, therefore we can rewrite

$$h\tilde{d}_k + \min \;\; E\{TC\} = \sum_{t=1,t\neq k}^{N} a\delta_t + \sum_{t=1,t\neq k-1}^{N} h\tilde{I}_t + h\tilde{I}_k \tag{5.32}$$

subject to,

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} \geq 0 \qquad t = 1, \ldots, N; t \neq k-1; t \neq k \tag{5.33}$$

$$\tilde{I}_k + \tilde{d}_k + \tilde{d}_{k-1} - \tilde{I}_{k-2} \geq 0 \tag{5.34}$$

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} > 0 \Rightarrow \delta_t = 1 \qquad t = 1, \ldots, N; t \neq k-1; t \neq k \tag{5.35}$$

$$\tilde{I}_k + \tilde{d}_k + \tilde{d}_{k-1} - \tilde{I}_{k-2} > 0 \Rightarrow \delta_{k-1} = 1 \tag{5.36}$$

$$\tilde{I}_t \geq b \left(\max_{j \in \{1..t\}} j \cdot \delta_j, t \right) \qquad t = 1, \ldots, N; t \neq k-1 \tag{5.37}$$

$$\tilde{I}_t \in \mathbb{Z}^+ \cup \{0\} \qquad t = 1, \ldots, N; t \neq k-1 \tag{5.38}$$

$$\delta_t \in \{0, 1\} \qquad t = 1, \ldots, N; t \neq k. \tag{5.39}$$

To summarize, we showed that constraint 5.7 for $t = k-1$ and $t = k$ can be expressed by Eq. 5.25, and similarly constraint 5.8 for $t = k-1$ and $t = k$ can be expressed by Eq. 5.26. Both these new constraints (5.25,5.26) are independent of $\tilde{I}_{k-1}$. Constraint 5.9 for $t = k-1$ becomes redundant. The new objective function (Eq. 5.31) reflects the consequences of constraint 5.24 and is independent of decision variable $\tilde{I}_{k-1}$. Therefore the whole model is now independent of decision variable $\tilde{I}_{k-1}$, whose value is a function of $\tilde{I}_k$ (Eq. 5.24).

Since the last model is independent of $\tilde{I}_{k-1}$ and δ_k, we now reduce it to an $(N-1)$-period model $\mathcal{R}$ through a change of variables, by merging periods $k-1$ and k and realizing the whole demand $\tilde{d'}_{k^*} = \tilde{d}_k + \tilde{d}_{k-1}$ in the new period k^*, where k^* covers the span $\{k-1, k\}$. In such a new model $\mathcal{R}$ the demand $\tilde{d'}_t$ in the other periods $t \in \{1, ..., k^*-1, k^*+1, ..., N-1\}$ is mapped as follows:

$$\tilde{d'}_t = \begin{cases} \tilde{d}_t, & t \in \{1, ..., k-2\} \\ \tilde{d}_{t+1}, & t \in \{k, ..., N-1\}. \end{cases}$$

Since the demand in periods k and $k-1$ of $\mathcal{P}$ is assumed to be normally distributed, the variance for the demand in the new period k^* of $\mathcal{R}$ is

$$\sigma'_{k^*} = \sqrt{\sigma_k^2 + \sigma_{k-1}^2}.$$

$\tilde{I'}_{k^*}$ in $\mathcal{R}$, that is the closing inventory levels in the new model, can be related to the respective closing inventory levels of periods k and $k-1$ in $\mathcal{P}$ using $\tilde{I}_k = \tilde{I'}_{k^*}$ and $\tilde{I}_{k-1} = \tilde{I'}_{k^*} + \tilde{d}_k$, which follow from Eq. 5.24 and the definition of k^*. The other closing inventory levels are mapped as follows:

$$\tilde{I'}_t = \begin{cases} \tilde{I}_t, & t \in \{1, ..., k-2\} \\ \tilde{I}_{t+1}, & t \in \{k, ..., N-1\}. \end{cases}$$

Notice that we only assumed $\delta_k = 0$, so $N-1$ binary decision variables are still unassigned. Therefore we have $\delta'_{k^*} = \delta_{k-1}$ (Eq. 5.26) and the following mapping for the remaining variables:

$$\delta'_t = \begin{cases} \delta_t, & t \in \{1, ..., k-2\} \\ \delta_{t+1}, & t \in \{k, ..., N-1\}, \end{cases}$$

where δ'_t are the binary decision variables in $\mathcal{R}$. Eq. 5.31 states that in order to get a model equivalent to the initial one, we must apply a holding cost of $2h$ for the new period k^* in the objective function.

The last model presented can be therefore rewritten in terms of the new decision variables defined by this mapping. The resulting problem instance is $\mathcal{R}$

$$E\{TC\} = h\tilde{d}_k + \min \sum_{t=1}^{N-1} a\delta'_t + \sum_{t=1}^{N-1} h\tilde{I'}_t + h\tilde{I'}_{k^*} \tag{5.40}$$

subject to

$$\tilde{I'}_t + \tilde{d'}_t - \tilde{I'}_{t-1} \geq 0 \qquad t = 1, \ldots, N-1 \tag{5.41}$$

$$\tilde{I'}_t + \tilde{d'}_t - \tilde{I'}_{t-1} > 0 \Rightarrow \delta'_t = 1 \qquad t = 1, \ldots, N-1 \tag{5.42}$$

$$\tilde{I'}_t \geq b \left(\max_{j \in \{1..t\}} j \cdot \delta'_j, t \right) \qquad t = 1, \ldots, N-1 \tag{5.43}$$

$$\tilde{I'}_t \in \mathbb{Z}^+ \cup \{0\}, \quad \delta'_t \in \{0, 1\} \qquad t = 1, \ldots, N-1. \tag{5.44}$$

It is trivial to recursively extend this reasoning to the case of consecutive periods with δ_k set to zero. This process necessarily ends when we reach an $i < k$ where $\delta_i = 1$ or $\delta_i \in \{0, 1\}$. Furthermore $\delta_1 = 1$, since without loss of generality we assume an initial null inventory and an initial demand greater than zero,

t	1	2	3	4	5	6	7	8	9
$i, \ldots, j$	$1, 2$	3	$4, 5$	$6, 7$	$8, 9$	10	$11, 12$	13	$14, 15$
$\tilde{d}_t$	73	128	208	208	192	88	94	181	96
σ_t	24.3	42.6	49.3	60.6	55.4	29.3	22.5	60.3	23.5
h_t	2	1	2	2	2	1	2	1	2
t	10	11	12	13	14				
$i, \ldots, j$	16	$17, 18, 19$	$20, 21$	22	$23, 24$				
$\tilde{d}_t$	88	52	209	190	195				
σ_t	29.3	13.7	64.2	63.3	55.3				
h_t	1	3	2	1	2				

Table 5.6: Reduced problem instance built as described in Step 1. For every period t in the new instance $\mathcal{R}$, $i, \ldots, j$ denotes the span covered in the original problem $\mathcal{P}$

i	$Dom(I'_i)$	i	$Dom(I'_i)$
$1 : \{1, 2\}$	$\{\underline{40}\}$	$8 : \{13\}$	$\{\underline{99}\}$
$2 : \{3\}$	$\{\underline{70}\}$	$9 : \{14, 15\}$	$\{\underline{39}\}$
$3 : \{4, 5\}$	$\{\underline{81}\}$	$10 : \{16\}$	$\{\underline{88}, 143\}$
$4 : \{6, 7\}$	$\{\underline{100}\}$	$11 : \{17, 18, 19\}$	$\{23, \underline{36}, 91\}$
$5 : \{8, 9\}$	$\{\underline{91}\}$	$12 : \{20, 21\}$	$\{\underline{106}\}$
$6 : \{10\}$	$\{\underline{88}\}$	$13 : \{22\}$	$\{\underline{104}\}$
$7 : \{11, 12\}$	$\{\underline{37}\}$	$14 : \{23, 24\}$	$\{\underline{91}\}$

Table 5.7: Effect of pre-processing method I in [92] on the smaller instance with merged periods, underlined figures are closing inventory levels of the optimal policy

therefore we always fix a replenishment in the first period.

□

Example 5.3.2. We now refer to the same instance analyzed for the example in Section 5.3.1. When the partial solution given in Table 5.4 is considered, a reduced problem instance can be built as described in Step 1. This instance is shown in Table 5.6. We applied pre-processing method I in [92] to this instance as stated in Step 2. Note that this is equivalent to applying our cost-based filtering method presented in Section 5.3.1 when in the given partial solution no decision variable has been assigned to a value. The reduced domains are shown in Table 5.7. From the reduced domains in Table 5.7, by applying Step 3, we can compute the reduced domain for the original problem instance. These domains are shown in Table 5.8. The two presented methods are incomparable, in fact this method prunes more values in period 6 while the former one prunes more values in period 16. ◇

i	$Dom(I_i)$	i	$Dom(I_i)$
1	$\{\underline{40}\}$	13	$\{\underline{99}\}$
2	$\{\underline{40}\}$	14	$\{\underline{73}\}$
3	$\{\underline{70}\}$	15	$\{\underline{39}\}$
4	$\{\underline{173}\}$	16	$\{\underline{88}, 143\}$
5	$\{\underline{81}\}$	17	$\{73, \underline{86}, 141\}$
6	$\{\underline{128}\}$	18	$\{63, \underline{76}, 131\}$
7	$\{\underline{100}\}$	19	$\{23, \underline{36}, 91\}$
8	$\{\underline{119}\}$	20	$\{\underline{123}\}$
9	$\{\underline{91}\}$	21	$\{\underline{106}\}$
10	$\{\underline{88}\}$	22	$\{\underline{104}\}$
11	$\{\underline{94}\}$	23	$\{\underline{123}\}$
12	$\{\underline{37}\}$	24	$\{\underline{91}\}$

Table 5.8: Reduced domains of the original instance obtained through the mapping proposed, underlined figures are closing inventory levels of the optimal policy

5.4 Cost-based filtering by relaxation

The CP model as described so far suffers from a lack of tight bounds on the objective function. In this section we recall a relaxation for our model originally proposed by Tarim in [86]. By means of this relaxation we will introduce a novel approach to compute a locally optimal solution or a valid lower bound at each node of the search tree.

It should be noted that the relaxation as presented in [86] does not take into account a given partial solution if this is available. As we will show this extension is not trivial, especially if we aim to take into account a partial assignment involving both δ_t and $\tilde{I}_t$ decision variables.

Given a problem instance, Tarim's approach adopts a greedy algorithm to solve a relaxed problem instance. This way a replenishment plan (assignment for the δ_t and I_t variables) is generated. Once this replenishment plan is available, it is possible to characterize if it is also feasible with respect to the original problem. If so, the respective computed cost is optimal for the original problem. Otherwise, if the replenishment plan is infeasible with respect to the original problem, the computed cost is a valid lower bound for the optimal solution cost of the original problem.

5.4.1 Tarim's relaxation

We shall now describe Tarim's relaxation in details. The core observation consists in the fact that the CP model proposed in Section 5.2 can be reduced to a **shortest path problem** if we relax inventory conservation constraints (5.7,5.8) for replenishment periods only. That is for each possible pair of replenishment cycles $\langle T(i,k-1), T(k,j)\rangle$ where $i,j,k \in \{1,\ldots,N\}$ and $i < k \leq j$, we do not consider the relationship between the opening inventory level of $T(k,j)$ and the closing inventory level of $T(i,k-1)$. This corresponds to allowing negative replenishments (Fig. 5.4 - a), or the ability to sell stock back to the supplier. Since the inventory conservation constraint is now relaxed between replenishment cycles, each replenishment cycle can be now treated independently and its cost can be computed *a priori*. In fact, given a replenishment cycle $T(i,j)$, we recall that $b(i,j)$, as defined above, denotes the minimum buffer stock level required to satisfy a given service level constraint during the replenishment cycle $T(i,j)$. It directly follows that $\tilde{I}_j = b(i,j)$. Furthermore for each period $t \in \{i,\ldots,j-1\}$ the expected closing-inventory-level is $\tilde{I}_t = b(i,j) + \sum_{k=t+1}^{j} d_k$. Since all the $\tilde{I}_t$ for $t \in \{i,\ldots,j\}$ are known it is easy to compute the expected total cost for $T(i,j)$, which is by definition the sum of the ordering cost and of the holding cost components, $a + h\sum_{t=i}^{j} \tilde{I}_t$. We now have a set $\mathcal{S}$ of $N(N+1)/2$ possible different replenishment cycles and the respective costs. Our new problem is to find an optimal set $\mathcal{S}^* \subset \mathcal{S}$ of consecutive disjoint replenishment cycles that covers our planning horizon at the minimum cost.

It should be noted that, from the characterization of the optimal policy for the deterministic inventory/production problem given by Wagner and Whitin [96], the optimal solution of this relaxation is always feasible for the original problem if buffer stocks are all zero and therefore we are solving a deterministic problem. In fact we recall that, as stated in [96] in the search for the optimal policy for the deterministic production/inventory problem it is sufficient to consider programs in which at period t one does not both place an order and bring in inventory (i.e. zero-inventory ordering property). It directly follows that every relaxed inventory conservation constraint is trivially satisfied under a deterministic setting, as in an optimal solution the closing inventory level at the end of each replenishment cycle

must be zero.

5.4.2 Tarim's relaxation as a shortest path problem

We shall now show that the optimal solution to this relaxation is given by the shortest path in a graph from a given initial node to a final node where each arc represents a replenishment cycle cost. If N is the number of periods in the planning horizon of the original problem, we introduce $N+1$ nodes. Since we assume, without loss of generality, that an order is always placed at period 1, we take node 1, which represents the beginning of the planning horizon, as the initial node. Node $N+1$ represents the end of the planning horizon. For each possible replenishment cycle $T(i, j-1)$ such that $i, j \in \{1, \ldots, N+1\}$ and $i < j$, we introduce an arc (i, j) with associated cost $c(i, j-1)$. Since we are dealing with a one-way temporal feasibility problem [96], when $i \geq j$, we introduce no arc. The connection matrix for such a graph, of size $N \times (N+1)$, can be built as shown in Table 5.9. By construction the cost of the shortest path from node 1 to node $N+1$ in

	1	2	$\ldots$	j	$\ldots$	$N+1$
1	$-$	$c(1,1)$	$\ldots$	$c(1,j-1)$	$\ldots$	$c(1,N)$
$\vdots$	$-$	$-$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
i	$-$	$-$	$-$	$c(i,j-1)$	$\ldots$	$c(i,N)$
$\vdots$	$-$	$-$	$-$	$-$	$\ddots$	$\vdots$
N	$-$	$-$	$-$	$-$	$-$	$c(N,N)$

Table 5.9: Shortest Path Problem Connection matrix

the given graph is a valid lower bound for the original problem, as it is a solution of the relaxed problem.

Solution mapping. It is easy to map the optimal solution for the relaxed problem, that is the set of arcs participating to the shortest path, to a solution for the original problem by noting that each arc (i, j) represents a replenishment cycle $T(i, j-1)$. By the definition of replenishment cycle $T(i, j-1)$, $\delta_i = 1$ and $\delta_t = 0$, for $t = i+1, \ldots, j-1$. The set of arcs in the optimal path uniquely identifies a set of disjoint replenishment cycles, that is a replenishment plan (assignment for

δ_t decision variables). Furthermore for each period $t \in \{i, \ldots, j-1\}$ in cycle $T(i, j-1)$ we already showed that all the expected closing-inventory-levels $\tilde{I}_t$, $t \in \{i, \ldots, j-1\}$, are known. This produces a complete assignment for decision variables in our model. The feasibility of such an assignment with respect to the original problem can be checked by verifying that it satisfies every relaxed constraint, that is no negative expected order quantity is scheduled.

Shortest path algorithm. To find a shortest path in the given graph we use a modified Dijkstra's algorithm that finds a shortest path in $O(n^2)$ time, where n is the number of nodes in the graph. Details on efficient implementations of Dijkstra's algorithm can be found in [77]. Usually Dijkstra's algorithm [77] does not apply any specific rule for labeling when ties are encountered in sub-path lengths. This non-deterministic labeling may produce a loss of optimal solutions if decision variable domains are pre-processed as described in [92]. In fact pre-processing Method I in [92] relies upon an upper-bound for optimal replenishment cycle length. When a replenishment period $i \in \{1, ..., N\}$ is considered, it looks for the lowest $j \in \{i, \ldots, N\}$ after which it is no longer optimal to schedule the next replenishment. This means that, if more policies that share the same expected cost exist, only the one that has shorter, and obviously more, replenishment cycles will be preserved by Method I. Therefore, when the algorithm is implemented in this filtering approach, we need to introduce a specific rule for node selection in order to make sure that, when more optimal policies exist, our modified algorithm will always find the one that has the highest possible number of replenishment cycles (i.e. the shortest path with the highest possible number of arcs). Since there is a complete order among nodes, we can easily implement this rule in the labeling action by always choosing as ancestor the node that minimizes the distance from the source and that has the highest index. The pseudo-code for the proposed modified Dijkstra's algorithm can be found in Appendix 5.7.3.

5.4.3 Cost-based filtering

So far we described a known possible way to relax the CP model proposed in Section 5.2. We also proposed a novel Dijkstra's algorithm implementation that

makes the relaxation in [86] compatible with the pre-processing methods in [92]. The relaxation described can be seen as a state space relaxation, where we define a new problem with a number of states polynomially bounded in the original problem input. A lower bound for the optimal solution cost is then obtained by solving a Shortest Path Problem in the state space graph. We will now show a novel approach to exploit this lower bound in an *optimization oriented global constraint*. A detailed discussion on state space relaxation and optimization oriented global constraints can be found in [33].

Partial assignments for δ_k decision variables

$\delta_k = 0$**:** Let us consider the graph built as described in Tarim's relaxation. If in a given partial solution a decision variable δ_k, $k \in \{1, \ldots, N\}$ has been already set to 0, then we can remove from the graph every inbound arc to node k and every outbound arc from node k. This prevents node k from being part of the shortest path, and hence prevents period k from being a replenishment period. By applying Dijkstra's algorithm to this modified graph the cost of the shortest path will provide a valid lower bound for the cost of an optimal solution incorporating the decision $\delta_k = 0$. Furthermore, as seen above, Dijkstra's algorithm will also provide an assignment for decision variables. If this assignment is feasible for the original problem, then it is optimal with the respect to the decision $\delta_k = 0$.

$\delta_k = 1$**:** On the other hand, if in a given partial solution a decision variable δ_k, $k \in \{1, \ldots, N\}$ has been already set to 1, then we can remove from the graph every arc connecting a node i to a node j, where $i < k < j$. This forces the shortest path to pass through node k, and hence forces period k to be a replenishment period. By applying Dijkstra's algorithm to this modified graph the cost of the shortest path will provide a valid lower bound for the cost of an optimal solution incorporating the decision $\delta_k = 1$. Furthermore, as seen above, Dijkstra's algorithm will also provide an assignment for decision variables. If this assignment is feasible for the original problem, then it is optimal with the respect to the decision $\delta_k = 1$.

We have shown how to act when each of the possible cases, $\delta_k = 1$ and $\delta_k = 0$,

is encountered. It is now possible at any point of the search in the decision tree to apply this relaxation and compute a valid lower bound or a solution that is optimal with respect to the given partial assignment.

Partial assignments for $\tilde{I}_k$ decision variables

It is also possible to extend this cost-based filtering method by considering not only the δ_k variable assignments, but also the $\tilde{I}_k$ variable assignments. In fact, when the cost of a given replenishment cycle $T(i, j-1)$ (arc (i, j) in the matrix) is computed, it is also possible to consider the current assignments for the closing inventory levels $\tilde{I}_k$ in the periods of this cycle. Since all the closing inventory levels of the periods within a replenishment cycle are linearly dependent ($\delta_k = 0 \rightarrow \tilde{I}_k + \tilde{d}_k - \tilde{I}_{k-1}$=0), given an assignment for a decision variable $\tilde{I}_k$ we can easily compute all the other closing inventory levels in the cycle by using $\tilde{I}_k - \tilde{d}_k - \tilde{I}_{k-1} = 0$, which is the inventory conservation constraint when no order is placed in period k. When the closing inventory levels in a replenishment cycle $T(i, j-1)$ are known it is easy to compute the overall cost associated to this cycle as seen above. We can therefore associate to arc (i, j) the highest cost that is produced by a current assignment for the closing inventory levels $\tilde{I}_k$, $k \in \{i, \ldots, j-1\}$. If no variable has been assigned yet, we simply use the minimum possible cost $c(i, j-1)$ which we defined above.

5.5 Experimental results

This section is organized as follows. Firstly we will consider a particularly hard instance built by adding random elements on a seasonal demand. We will use this instance to gauge the effectiveness of each filtering method we proposed. Furthermore we will also analyze how the proposed methods perform when they are combined together. Secondly we will compare our method with the state-of-the-art results presented in [92]. Thirdly we will present extensive tests to show the effectiveness of our domain filtering methods with respect to a pure CP approach enhanced with the pre-processing methods presented by Tarim and Smith.

All experiments presented here were performed on an Intel(R) Centrino(TM) CPU 1.50GHz with 500Mb RAM. The solver used for our test is Choco [58], an open-source solver developed in Java.

The heuristic used for the selection of the variable is the usual min-domain/max-degree heuristic. Decision variables have different priorities in the heuristic: the δ_k have higher priority than the $\tilde{I}_k$. The value selection heuristic chooses values in increasing order of size.

In what follows we will refer to the filtering methods presented as follows: Method I (Section 5.3.1), Method II (Section 5.3.2), Method III (Section 5.4). Since Method II can be in principle applied in conjunction with any sound domain reduction method, in all the experiments here presented the domain reduction applied with Method II is pre-processing method II presented in Tarim and Smith [92]. We only apply one pre-processing method since experimentally no improvement was noticed in term of explored nodes and running time when both the methods were used in conjunction as shown in [92].

5.5.1 Effectiveness of filtering methods

A single problem is considered and the period demands are listed in Figure 5.5. In each test we assume an initial null inventory level and a normally distributed

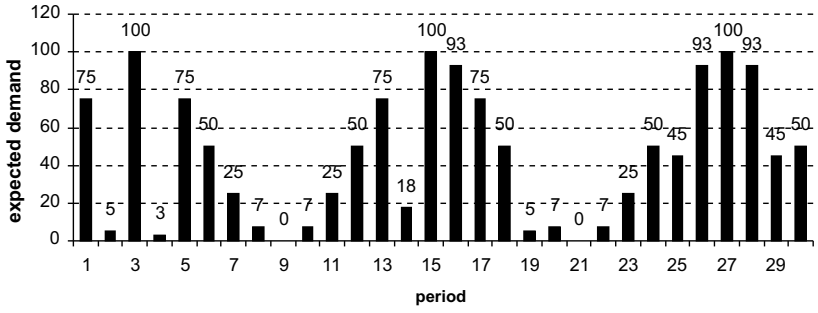


Figure 5.5: Expected demand values

demand for every period with a coefficient of variation $\sigma_t/\tilde{d}_t = 1/3$ for each $t \in \{1, \ldots, N\}$, where N is the length of the planning horizon considered. The ordering cost ranges in the following set $\{40, 80, 160, 320\}$. The holding cost is 1. Our tests consider two different service levels $\alpha = 0.95$ ($z_{\alpha=0.95} = 1.645$)

		No Filt.		Method I		Method II		Method III		Combined	
α	a	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec
0.95	40	127	1.85	96	1.64	96	1.43	120	1.30	70	1.12
	80	2994	30	1449	16	2586	23	82	1.02	63	0.97
	160	–	–	–	–	–	–	133	1.81	108	1.65
	320	–	–	–	–	–	–	4	0.09	4	0.09
0.99	40	261	3.27	198	4.24	202	2.52	253	2.84	165	2.57
	80	1234	11	611	7.54	1138	10.7	317	2.66	221	2.61
	160	–	–	–	–	–	–	168	2.15	84	1.31
	320	–	–	–	–	–	–	1	0.09	1	0.10

Table 5.10: Filtering methods compared in terms of explored nodes ("Nod") and run time in seconds ("Sec"). Symbol "–" means that an optimal solution has not be found within the given limit of 60 secs

and $\alpha = 0.99$ ($z_{\alpha=0.99} = 2.326$). In Table 5.10 we compare the effectiveness of each filtering method, when used to augment the CP model enhanced by the pre-processing methods in [92]. The performances achieved by the CP approach enhanced with the pre-processing methods are shown in column "No Filt.". The performances achieved when the filtering methods are all added to the model are shown in column "Combined". In the presented table we can see that Method I and Method II do not perform well when they are used alone. This is again due to the lack of good bounds during the search process. Method III instead is very effective even when it is used alone and especially for high ordering costs, when the contribution of the filtering due to the computed bounds is critical. Nevertheless when the three methods are combined for all the eight instances presented performances are improved both in terms of running time and explored nodes.

5.5.2 Comparison with state-of-the-art results

In this section we compare results obtained with our approach with the state-of-the-art results presented in [92].

A single problem is considered and the period demands are generated from seasonal data with no trend: $\tilde{d}_t = 50[1 + \sin(\pi t/6)]$. In addition to the "no trend" case (P1) we also consider three others:

(P2) positive trend case, $\tilde{d}_t = 50[1 + \sin(\pi t/6)] + t$

(P3) negative trend case, $\tilde{d}_t = 50[1 + \sin(\pi t/6)] + (52 - t)$

		$a = 400$				$a = 800$			
		Filt.		Tarim & Smith		Filt.		Tarim & Smith	
	Horizon	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec
P1	50	1	0.30	–	–	3	0.10	–	–
	48	1	0.09	–	–	3	0.10	30795352	10100
	46	1	0.09	43721791	12200	3	0.09	8763280	2840
	44	1	0.09	36976882	9700	3	0.01	6896956	2110
P2	44	1	0.09	–	–	4	0.10	–	–
	42	1	0.09	–	–	4	0.10	60884565	15600
	40	1	0.29	–	–	4	0.17	22281926	5590
	38	1	0.09	35848309	6820	4	0.10	7978185	1880
P3	42	1	0.09	–	–	3	0.10	–	–
	40	1	0.09	–	–	3	0.10	55138095	13300
	38	1	0.09	61438266	11300	3	0.10	19600638	4510
	36	1	0.09	24256921	4150	3	0.10	6501541	1510
P4	44	1	0.09	–	–	4	0.09	–	–
	42	1	0.10	–	–	4	0.11	39668737	10700
	40	1	0.09	–	–	4	0.10	18004555	4690
	38	1	0.09	32076069	6680	4	0.09	6093007	1520

Table 5.11: Comparison with the state-of-the-art results in [92] ("Tarim & Smith"). "Filt." indicates that Tarim & Smith's model is augmented with our filtering methods. Symbol "–" means that an optimal solution has not been found within the given limit of 5 hours

(P4) life-cycle trend case, $\tilde{d_t} = 50[1 + \sin(\pi t/6)] + \min(t, 52 - t)$

In each test we assume a coefficient of variation $\sigma_t/\tilde{d_t} = 1/3$ for each $t \in \{1, \ldots, N\}$, where N is the length of the considered planning horizon. As in Tarim and Smith tests are performed using two different ordering cost values $a \in \{400, 900\}$. The holding cost used in these tests is $h = 1$ per unit per period. Our tests consider a service levels $\alpha = 0.95$ ($z_{\alpha=0.95} = 1.645$).

In Table 5.11 we can observe the improvement of several orders of magnitude brought by our domain filtering techniques. Experiments in [92] employed OPL Studio 3.7 (ILOG Solver 6.0, ILOG Cplex 9.0) used with its default settings. Note that the hardware used for these experiments is comparable to the one used for ours.

5.5.3 More extensive tests

In this section we show the effectiveness of our approach by comparing the computational performance of the state-of-the-art CP model with that obtained by our approach.

We refer again to (P1), (P2), (P3) and (P4) as defined above. We performed tests using four different ordering cost values $a \in \{40, 80, 160, 320\}$ and two different $\sigma_t/\tilde{d}_t \in \{1/3, 1/6\}$. The planning horizon length takes even values in the range $[24, 50]$ when the ordering cost is 40 or 80 and $[14, 24]$ when the ordering cost is 160 or 320. The holding cost used in these tests is $h = 1$ per unit per period. Our tests also consider two different service levels $\alpha = 0.95$ ($z_{\alpha=0.95} = 1.645$) and $\alpha = 0.99$ ($z_{\alpha=0.99} = 2.326$).

In our test results a time of 0 means that the Dijkstra algorithm proved optimality at the root node. A header "Filt." means that we are applying our cost-based filtering methods, and "No Filt." means that we solve the instance using only the CP model and the pre-processing methods. Tables 5.12, 5.13, 5.14 and 5.15 compare the performance of the state-of-the-art CP model, implemented in Choco, with that of our new methods.

		$\sigma_t/d_t = 1/3$								$\sigma_t/d_t = 1/6$							
		$\alpha = 0.95$				$\alpha = 0.99$				$\alpha = 0.95$				$\alpha = 0.99$			
		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.	
a	N	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec
40	40	28	0.9	106	2.9	38	0.8	249	6.4	34	0.7	574	16	10	0.1	192	6.4
	42	28	0.6	95	2.8	38	0.8	233	5.9	34	0.7	582	14	10	0.1	196	5.4
	44	29	0.6	133	4.9	47	1.1	266	8.3	35	0.7	884	25	11	0.2	285	9.0
	46	30	0.6	192	7.8	64	1.6	484	18	45	1.0	3495	120	13	0.2	813	30
	48	43	0.9	444	19	72	2.1	1024	41	53	1.1	5182	190	13	0.2	1208	47
	50	43	1.0	444	20	72	2.2	1024	44	53	1.2	4850	200	13	0.2	1208	51
80	40	43	0.8	1742	78	13	0.2	557	15	16	0.2	9316	300	16	0.3	11276	440
	42	43	0.9	1703	60	13	0.2	530	13	17	0.3	17973	530	17	0.3	22291	690
	44	48	1.1	4810	210	14	0.2	980	25	19	0.4	38751	1400	20	0.4	50805	1600
	46	49	1.3	6063	340	16	0.3	2122	78	20	0.3	103401	4300	20	0.4	111295	4100
	48	67	2.0	20670	1400	17	0.3	5284	210	21	0.4	237112	12000	21	0.5	321998	15000
	50	67	2.2	18938	1300	17	0.4	5284	230	21	0.4	251265	13000	21	0.5	358174	17000
160	14	1	0.0	141	3.0	23	0.1	156	2.5	1	0.0	112	2.6	1	0.0	116	2.4
	16	1	0.0	277	9.0	35	0.2	182	5.1	1	0.0	238	6.7	1	0.0	235	6.8
	18	1	0.0	673	18	41	0.4	393	10	1	0.0	799	23	1	0.0	603	15
	20	1	0.0	3008	81	51	0.6	1359	21	1	0.0	2887	86	1	0.0	2820	75
	22	1	0.0	10620	260	57	0.6	7280	70	1	0.0	14125	380	1	0.0	10739	270
	24	1	0.0	61100	1500	153	1.8	31615	310	1	0.0	70996	1800	1	0.0	59650	1500
320	14	1	0.0	149	4.0	1	0.0	181	4.1	1	0.0	109	3.0	1	0.0	128	3.0
	16	1	0.0	335	11	1	0.0	361	12	1	0.0	246	8.7	1	0.0	284	9.3
	18	1	0.0	813	27	1	0.0	831	27	1	0.0	764	26	1	0.0	700	24
	20	1	0.0	2602	93	1	0.0	2415	81	1	0.0	2114	78	1	0.0	2291	82
	22	1	0.0	7434	260	1	0.0	7416	260	1	0.0	7006	260	1	0.0	6608	230
	24	1	0.0	49663	1600	1	0.0	49299	1500	1	0.0	39723	1400	1	0.0	43520	1500

Table 5.12: Test set P1

		$\sigma_t/d_t = 1/3$								$\sigma_t/d_t = 1/6$							
		$\alpha = 0.95$				$\alpha = 0.99$				$\alpha = 0.95$				$\alpha = 0.99$			
		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.	
a	N	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec
40	40	4	0.1	7	0.1	7	0.1	8	0.1	12	0.2	23	0.4	4	0.1	12	0.2
	42	4	0.0	7	0.1	7	0.2	8	0.1	12	0.2	23	0.4	4	0.1	10	0.1
	44	4	0.1	7	0.1	7	0.2	8	0.1	12	0.2	23	0.5	4	0.1	10	0.2
	46	4	0.1	7	0.1	7	0.3	8	0.2	12	0.2	23	0.5	4	0.1	10	0.2
	48	4	0.1	7	0.2	7	0.2	8	0.2	12	0.3	23	0.5	4	0.1	10	0.2
	50	4	0.1	7	0.2	7	0.2	8	0.2	12	0.3	23	0.6	4	0.1	10	0.2
80	40	18	0.3	4592	14	15	0.3	275	8.3	37	0.7	2565	63	32	0.7	1711	44
	42	18	0.4	4866	13	15	0.4	283	6.7	37	0.8	3027	67	32	0.7	2043	47
	44	18	0.4	5091	15	15	0.4	280	7.9	40	0.9	6024	160	37	0.9	4299	120
	46	23	0.5	5291	45	17	0.5	545	16	47	1.3	14058	410	39	1.1	10311	290
	48	23	0.6	5544	51	17	0.5	545	17	47	1.4	14058	440	39	1.2	10311	310
	50	23	0.6	5850	51	17	0.5	545	18	47	1.5	14079	470	39	1.3	10347	330
160	14	1	0.0	166	3.6	19	0.1	84	1.0	1	0.0	148	2.9	1	0.0	171	3.4
	16	30	0.2	154	4.3	19	0.1	65	1.2	1	0.0	329	8.6	1	0.0	383	10
	18	58	0.4	485	11	34	0.3	174	2.9	1	0.0	948	23	1	0.0	1056	27
	20	37	0.3	2041	35	37	0.4	707	7.9	1	0.0	4228	110	1	0.0	4730	120
	22	48	0.4	9534	120	32	0.3	2954	28	1	0.0	20438	500	1	0.0	23675	530
	24	65	0.7	30502	360	41	0.4	7787	87	1	0.0	71514	1800	1	0.0	83001	1900
320	14	1	0.0	238	5.6	1	0.0	278	6.4	1	0.0	166	3.7	1	0.0	191	4.5
	16	1	0.0	505	17	1	0.0	423	13	1	0.0	387	11	1	0.0	452	14
	18	1	0.0	1447	49	1	0.0	1208	40	1	0.0	1100	34	1	0.0	1268	40
	20	1	0.0	4792	156	1	0.0	4219	150	1	0.0	3992	130	1	0.0	4476	150
	22	1	0.0	20999	660	1	0.0	20417	610	1	0.0	15983	520	1	0.0	18663	600
	24	1	0.0	102158	3200	1	0.0	90398	2600	1	0.0	75546	2500	1	0.0	88602	2800

Table 5.13: Test set P2

		$\sigma_t/d_t = 1/3$								$\sigma_t/d_t = 1/6$							
		$\alpha = 0.95$				$\alpha = 0.99$				$\alpha = 0.95$				$\alpha = 0.99$			
		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.	
a	N	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec
40	40	2	0.0	5	0.0	2	0.0	4	0.0	6	0.1	9	0.2	2	0.0	5	0.0
	42	2	0.0	5	0.0	2	0.0	4	0.0	6	0.1	9	0.2	2	0.0	5	0.0
	44	3	0.0	7	0.1	3	0.0	6	0.1	7	0.1	14	0.3	3	0.0	7	0.1
	46	4	0.1	15	0.3	6	0.1	13	0.3	11	0.2	40	1.1	4	0.1	14	0.3
	48	4	0.1	15	0.3	6	0.1	13	0.3	14	0.3	56	1.8	4	0.1	25	0.6
	50	4	0.1	15	0.3	6	0.2	13	0.3	14	0.4	56	1.9	4	0.1	25	0.5
80	40	22	0.4	349	10	6	0.1	55	1.2	19	0.3	722	19	9	0.2	310	8.7
	42	22	0.4	354	8.6	6	0.1	53	1.2	22	0.4	1436	35	9	0.2	315	7.5
	44	24	0.6	571	17	7	0.1	88	2.4	27	0.6	3461	110	13	0.3	1053	31
	46	29	0.8	2787	90	9	0.2	258	8.1	36	0.9	10612	360	16	0.4	2881	94
	48	38	1.1	6803	240	9	0.2	385	12	47	1.3	28334	1100	22	0.6	7790	280
	50	38	1.1	6575	240	9	0.2	385	13	47	1.6	26280	1100	22	0.6	7371	280
160	14	7	0.0	23	0.2	8	0.0	16	0.1	15	0.1	53	0.6	9	0.0	29	0.3
	16	7	0.0	19	0.2	8	0.0	18	0.2	15	0.1	52	0.8	9	0.0	26	0.4
	18	9	0.1	42	0.5	10	0.0	30	0.3	21	0.1	149	2.2	12	0.1	87	1.2
	20	11	0.1	137	1.3	11	0.1	70	0.7	25	0.2	512	6.1	16	0.2	310	3.5
	22	21	0.2	376	4.0	21	0.2	221	2.3	31	0.4	1848	17	17	0.2	938	9.4
	24	32	0.4	995	11	30	0.4	543	6.3	43	0.5	4784	54	23	0.2	2471	30
320	14	1	0.0	253	4.2	1	0.0	232	3.8	1	0.0	310	4.4	1	0.0	217	3.4
	16	1	0.0	518	10	1	0.0	518	10	1	0.0	707	13	1	0.0	465	8.5
	18	1	0.0	1475	35	1	0.0	1170	26	1	0.0	1995	43	1	0.0	1416	33
	20	1	0.0	5342	140	1	0.0	4059	95	1	0.0	6678	160	1	0.0	5232	140
	22	1	0.0	21298	550	1	0.0	18065	440	1	0.0	25522	640	1	0.0	21756	560
	24	1	0.0	86072	2300	1	0.0	70969	1800	1	0.0	101937	2800	1	0.0	91358	2400

Table 5.14: Test set P3

		$\sigma_t/d_t = 1/3$								$\sigma_t/d_t = 1/6$							
		$\alpha = 0.95$				$\alpha = 0.99$				$\alpha = 0.95$				$\alpha = 0.99$			
		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.		Filt.		No Filt.	
a	N	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec	Nod	Sec
40	40	5	0.1	21	0.3	10	0.2	24	0.5	25	0.5	89	1.8	5	0.1	33	0.5
	42	5	0.1	18	0.3	10	0.2	21	0.4	25	0.5	91	2.0	5	0.1	31	0.5
	44	6	0.1	32	0.7	11	0.2	37	0.9	28	0.6	152	3.6	6	0.1	51	1.0
	46	7	0.1	83	2.0	16	0.4	93	2.4	42	1.0	474	12	7	0.1	126	2.8
	48	7	0.1	83	2.2	16	0.4	93	2.6	50	1.2	735	20	7	0.2	188	4.5
	50	7	0.1	83	2.3	16	0.4	93	2.8	50	1.4	735	22	7	0.2	188	4.9
80	40	39	0.7	1372	39	16	0.4	433	12	40	0.7	5098	130	33	0.7	2133	54
	42	39	0.8	1673	39	16	0.4	438	10	46	1.0	11452	270	33	0.8	2513	58
	44	43	1.0	2907	74	17	0.5	716	22	56	1.4	27184	780	46	1.3	8776	240
	46	51	1.3	13306	380	21	0.6	2178	73	75	1.9	77332	2600	55	1.6	22582	690
	48	69	1.8	32709	1000	21	0.6	3223	120	100	2.8	202963	7500	73	2.2	60115	2000
	50	69	1.9	31547	1100	21	0.7	3223	130	100	2.9	191836	7600	73	2.4	58171	2100
160	14	1	0.0	166	3.6	19	0.1	84	1.5	1	0.0	148	3.0	1	0.0	171	3.4
	16	30	0.2	154	4.3	19	0.1	65	1.6	1	0.0	329	8.7	1	0.0	383	10
	18	58	0.4	485	11	34	0.3	174	4.0	1	0.0	948	24	1	0.0	1056	27
	20	37	0.3	2041	34	37	0.4	707	11	1	0.0	4228	110	1	0.0	4730	120
	22	48	0.4	9534	120	32	0.3	2954	40	1	0.0	20438	510	1	0.0	23675	540
	24	65	0.7	30502	360	41	0.4	7787	130	1	0.0	71514	1800	1	0.0	83001	1900
320	14	1	0.0	238	5.5	1	0.0	278	8.7	1	0.0	166	3.7	1	0.0	191	4.5
	16	1	0.0	505	17	1	0.0	423	17	1	0.0	387	11	1	0.0	452	13
	18	1	0.0	1447	48	1	0.0	1208	57	1	0.0	1100	33	1	0.0	1268	40
	20	1	0.0	4792	160	1	0.0	4219	200	1	0.0	3992	130	1	0.0	4476	150
	22	1	0.0	20999	660	1	0.0	20417	860	1	0.0	15983	520	1	0.0	18663	600
	24	1	0.0	102158	3200	1	0.0	90398	3700	1	0.0	75546	2700	1	0.0	88602	2800

Table 5.15: Test set P4

When a=320, and often when a=160, the Dijkstra algorithm proves optimality at the root node so the other reduction methods are not exploited during search. This is a direct consequence of the fact that under high ordering cost values it is extremely rare that a solution for the relaxed problem violates some inventory conservation constraint. In fact since placing an order is expensive the optimal solution will try to cover several periods with a single order. Such an order requires a high order-up-to-level that typically exceeds the expected closing-inventory-level of the previous replenishment cycle. Therefore the solution of the relaxed problem solved by means of dynamic programming is usually feasible with respect to the original problem.

When $a \in \{40, 80\}$ Dijkstra is often unable to prove optimality at the root node, since the solution of the relaxed problem can easily violate inventory conservation constraints in the original problem under low ordering costs. This is due to the fact that the order-up-to-level for a replenishment cycle may easily be lower than the buffer stock levels held at the end of the former cycle. The main contribution brought by our relaxation in this situation consists in computing lower bounds during the search. Therefore in this case the domain reduction achieved with the other two filtering methods developed is critical in reducing the number of feasible values in the domain of expected closing-inventory-level decision variables. As shown in the experiments our approach can easily solve instances with up to 50 periods, both in terms of explored nodes and run time, for every combination of parameters we considered. In contrast, for the CP model both the run times and the number of explored nodes grow exponentially with the number of periods, and the problem becomes intractable for instances of significant size. In all cases our method explores fewer nodes than the pure CP approach, ranging from an improvement of one to several orders of magnitude. Apart from a few trivial instances on which both methods take a fraction of a second, this improvement is reflected in the run times.

5.6 Conclusions

It was previously shown [92] that CP is more natural than mathematical programming for expressing constraints for lot-sizing under the (R^n, S^n) policy, and leads

to more efficient solution methods. This paper further improves the efficiency of the CP-based approach by exploiting three forms of cost-based filtering. The wide test bed considered shows the effectiveness of our approach under many different parameter configurations and demand trends. The improvement reaches several orders of magnitude in almost every instance we analyzed. We are now able to solve to optimality problems of a realistic size, in times of less than a second and often without search, since the bounds produced by our DP relaxation proved to be very tight in a large amount of instances. In future work we aim to extend our model to new features such as lead-time for orders and capacity constraints for the inventory.

5.7 Appendix

5.7.1 Considering a unit production cost p

The stochastic programming formulation given can be extended to consider a unit production cost p as follows

$$\min \; E\{TC\} = \int_{d_1}\int_{d_2}\dots\int_{d_N}\sum_{t=1}^{N}(a\delta_t + h\cdot\max(I_t,0) + p\cdot Q_t) \\ g_1(d_1)g_2(d_2)\dots g_N(d_N)\mathrm{d}(d_1)\mathrm{d}(d_2)\dots\mathrm{d}(d_N) \tag{5.45}$$

subject to Constraint (5.2), (5.3), (5.4) and (5.5), for $t = 1,\dots,N$. The given objective function (5.45) can be rewritten as

$$E\{TC\} = p\cdot K + \min \int_{d_1}\int_{d_2}\dots\int_{d_N} p\cdot I_N + \sum_{t=1}^{N}(a\delta_t + h\cdot\max(I_t,0)) \\ g_1(d_1)g_2(d_2)\dots g_N(d_N)\mathrm{d}(d_1)\mathrm{d}(d_2)\dots\mathrm{d}(d_N) \tag{5.46}$$

where $K = \int_{d_1}\int_{d_2}\dots\int_{d_N}\sum_{t=1}^{N} d_t \;\; g_1(d_1)g_2(d_2)\dots g_N(d_N)\mathrm{d}(d_1)\mathrm{d}(d_2)\dots\mathrm{d}(d_N)$. For further details on this transformation the reader may refer to [71, 90], where a similar transformation is described in details for the stochastic inventory control

problem under a penalty cost scheme. Intuitively, objective function (5.46) shows that the effect of the unit production cost p can be decomposed in a constant factor $p \cdot K$ and in a variable factor $p \cdot I_N$ that depends on the very last closing-inventory-level planned. The deterministic equivalent CP approach is

$$E\{TC\} = p \sum_{t=1}^{N} \tilde{d}_t + \min \quad \left[p \cdot \tilde{I}_N + \sum_{t=1}^{N} \left(a\delta_t + h\tilde{I}_t \right) \right] \tag{5.47}$$

subject to Constraint (5.7), (5.8), (5.9) and (5.10), for $t = 1 \ldots N$. It directly follows that the variable effect of the unit production cost p is reflected only on the cost of the very last replenishment cycle scheduled. The cost-based filtering method presented in Section 5.3.2 is independent of the considerations presented here. It remains sound under a unit production cost if the associated pre-processing method can consider this cost. The pre-processing methods in [92] and the cost-based filtering method in Section 5.3.1 can be extended to consider a unit production cost p by replacing the definition given in Eq. (5.15) for the cost $c(i, j)$ of a replenishment cycle $T(i, j)$ as follows:

$$\widehat{c}(i,j) = \begin{cases} c(i,j) & \text{if} \quad j \neq N \\ p \cdot b(i,j) + c(i,j) & \text{if} \quad j = N. \end{cases} \tag{5.48}$$

The cost-based filtering in Section 5.4 in a similar manner applies to the case where a unit production cost p is considered if, when the connection matrix for the graph constructed is built, $c(i, j)$ is replaced by $\widehat{c}(i, j)$ as just described.

5.7.2 Proof: Replenishment cycle length bound

By using the definition of $c(i, j)$ we can rewrite Eq. 5.17 as

$$a + h(k - i + 1)b(i,k) + h \sum_{t=1}^{k} (t - i)\tilde{d}_t + a + h(j - k + 1)b(k + 1, j + 1) + h \sum_{t=k+1}^{j+1} (t - k - 1)\tilde{d}_t \leq a + h(j - i + 2)b(i, j + 1) + h \sum_{t=i}^{j+1} (t - i)\tilde{d}_t \tag{5.49}$$

which can be simplified to

$$\frac{a}{h} - \sum_{t=k+1}^{j+1} (k+1-i)\tilde{d}_t \leq (j-k+1)\left[b(i,j+1) - b(k+1,j+1)\right] + (k-i+1)\left[b(i,j+1) - b(i,k)\right]. \tag{5.50}$$

We now want to prove that if $p > j+1$, then $\exists k+1 \in \{i+1, j\}$ s.t. $c(i,k) + c(k+1,p) \leq c(i,p) \wedge b(i,k) \leq R(k+1,p)$. We can rewrite this condition as we did before and therefore obtain an expression similar to Eq. 5.50, that is

$$\frac{a}{h} - \sum_{t=k+1}^{p} (k+1-i)\tilde{d}_t \leq (p-k)\left[b(i,p) - b(k+1,p)\right] + (k-i+1)\left[b(i,p) - b(i,k)\right]. \tag{5.51}$$

We now subtract both the left and the right term of Eq. 5.50 from Eq. 5.51. Thus we get

$$\begin{aligned} -\sum_{t=j+2}^{p} (k+1-i)\tilde{d}_t + (j-k+1)\left[b(i,j+1) - b(k+1,j+1)\right] + \\ (k-i+1)\left[b(i,j+1) - b(i,k)\right] \leq (p-j-1)\left[b(i,p) - b(k+1,p)\right] + \\ (j-k+1)\left[b(i,p) - b(k+1,p)\right] + (k-i+1)\left[b(i,p) - b(i,k)\right], \end{aligned} \tag{5.52}$$

by omitting the term $-\sum_{t=j+2}^{p}(k+1-i)\tilde{d}_t$ to save space and rearranging the other terms we obtain

$$\begin{aligned} (j-k+1)\left[b(k+1,p) - b(k+1,j+1)\right] \leq \\ (j-i+2)\left[b(i,p) - b(i,j+1)\right] + (p-j-1)\left[b(i,p) - b(k+1,p)\right], \end{aligned} \tag{5.53}$$

we change name to the coefficients

$$\begin{aligned} A \cdot b(k+1,p) - A \cdot b(k+1,j+1) \leq \\ B \cdot b(i,p) - B \cdot b(i,j+1) + C \cdot b(i,p) - C \cdot b(k+1,p) \end{aligned} \tag{5.54}$$

and finally

$$(A+C)\cdot b(k+1,p) - A \cdot b(k+1,j+1) \leq (B+C)\cdot b(i,p) - B \cdot b(i,j+1), \tag{5.55}$$

where $A + C = p - k$ and $B + C = p - i + 1$. Reinserting the omitted term we obtain Eq. 5.18. Since $b(i,k) \leq R(k+1, j+1)$, it also follows that $b(i,k) \leq R(k+1,p)$. Therefore, under the given conditions, it is never optimal to cover the span $\{i,\ldots,p\}$, $p > j$ by using a single replenishment cycle $T(i,p)$. Hence the optimum period $k+1$ for the next replenishment after the one scheduled in period i lies in the span $\{i+1,\ldots,j+1\}$ and it cannot be after $j+1$.

5.7.3 Modified Dijkstra's Shortest Path Algorithm

We will use a modified implementation of Dijkstra's Shortest Path Algorithm in order to enhance performances and make our relaxation compatible with Method I in [92]. Dijkstra's strategy relies on the following well known Shortest Path Theorem, which holds for any directed acyclic graph

Theorem 5.7.1 (Shortest Path Theorem)**.** *If P is the shortest path from node u to node v and if P passes through node z, then P is made up by the shortest path Q_1 from u to z and by the shortest path Q_2 from z to v.*

Since we are solving a problem that implies a one-way temporal feasibility, as Wagner and Whitin notice in [96], half of our connection matrix will be set to ∞. Therefore any instance of size N can be solved in $N(N+1)/2$ steps taking this fact into account during the computation as we will see.

Let G be a directed acyclic graph $\langle V, A\rangle$, where V is a set of N numbered vertices $\{v_1, ..., v_N\}$ and A is a set of arcs among these nodes. Let W be a square matrix representing the cost related to each arc that appears in A. Let v_1 be the source we are computing shortest paths from. Let $d[v_i]$ be a label for any vertex $v_i \in V$, and $a[v_i]$ the index of the ancestor of node $v_i \in V$ in the shortest path. At the end of the computation $d[v_i]$ represents the shortest distance from the source v_1 to the vertex v_i. It is also possible to find every vertex in the shortest path from v_i to v_1 following in a recursive fashion the chain of indexes that starts with $a[v_i]$. In particular we will be interested in the shortest path from v_N to v_1, which is the one that covers our planning horizon. The complete code is shown in Algorithm 6. In order to reduce steps to $N(N+1)/2$ we introduced $j > i$ as a precondition for the execution of Procedure $Relax$(v_i,v_j,W). Notice also that in order to make

the algorithm compatible with filtering methods in [92] some checks on vertex indexes have been introduced. In particular in Procedure $Relax$(v_i,v_j,W) when two or more paths exist with the same distance from v_1 we always choose the ancestor v_i that has the highest index i. The reason we do this is related to the way pre-processing Method I in [92] filters values in decision variables domain. In fact, when a replenishment period i, $i \in \{1, ..., N\}$ is considered, such a method looks for the lowest j s.t. $j \geq i$ after which it is not longer optimal to schedule the next replenishment. This means that, if more policies that share the same expected cost exist, only the one that has shorter, and obviously more, replenishment cycles will be preserved by Method I, while values that are feasible with respect to other policies equally costly may be pruned. So we introduced the described checks on vertex indexes in order to make sure that, when more optimal policies exist, our modified algorithm will always find the one that has the highest possible number of replenishment cycles (i.e. the shortest path with the highest possible number of arcs).

Acknowledgements

This work was supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain-Driven Research (CTVR) and Grant No. 00/PI.1/C075.

Algorithm 6: Modified Shortest Path Algorithm

input : G, W, v_1
output: d, a

begin
  $Initialize(G, v_1)$
  Let $d[v_i]$ be the shortest path from v_i to v_1
  Insert all vertices in G in a priority queue Q
  while Q *is not empty* **do**
    extract v_i s.t. $d[v_i]$ is minimum
    for *each vertex* v_j *adjacent to* v_i *s.t.* $j > i$ **do**
      $Relax(v_i, v_j, W)$
end

Procedure `Initialize(`G,v_1`)`

begin
  for *each vertex* v_i *in* G **do**
    set $d[v_i]$ to $W(v_1, v_i)$
    set $a[v_i]$ to 1
  set $d[v_1]$ to 0
end

Procedure `Relax(`v_i,v_j,W`)`

begin
  if $d[v_j] > d[v_i] + W(v_i, v_j)$ **then**
    set $d[v_j]$ equal to $d[v_i]$ + $W(v_i, v_j)$
    set $a[v_j]$ equal to i
  else
    if $d[v_j] == d[v_i] + W(v_i, v_j)$ *AND* $i > a[v_j]$ **then**
      set $a[v_j]$ equal to i
end

Chapter 6

Paper V: Constraint Programming for Stochastic Inventory Systems under Shortage Cost

R. Rossi, S. A. Tarim, B. Hnich and S. Prestwich

Abstract

One of the most important policies adopted in inventory control is the (R,S) policy (also known as the "replenishment cycle" policy). Under the non-stationary demand assumption the (R,S) policy takes the form (R_n,S_n) where R_n denotes the length of the n^{th} replenishment cycle, and S_n the corresponding order-up-to-level. Such a policy provides an effective means of damping planning instability and coping with demand uncertainty. In this paper we develop a constraint programming approach able to compute optimal (R_n,S_n) policy parameters under stochastic demand, ordering, holding and shortage costs. We use the optimal solutions to analyze the quality of the solutions provided by an existing approximate mixed integer programming approach that exploits a piecewise linear approximation for the cost function. Furthermore we show how in our model it is possible to exploit the convexity of the cost-function during the search to dynamically compute bounds during the search and perform cost-based filtering.[†]

[†]This paper is an extended version of the work presented in [71]

6.1 Introduction

Much of the inventory control literature concerns the computation of optimal replenishment policies under demand uncertainty. One of the most important policies adopted is the (R,S) policy (also known as the *replenishment cycle* policy). A detailed discussion on the characteristics of (R,S) can be found in (de Kok [22]). In this policy a replenishment is placed every R periods to raise the inventory position to the order-up-to-level S. This provides an effective means of damping planning instability – deviations in planned orders, also known as *nervousness* (de Kok and Inderfurth [23], Heisig [44]) – and coping with demand uncertainty. As pointed out by (Silver et al. [81], pp. 236–237), (R,S) is particularly appealing when items are ordered from the same supplier or require resource sharing. In these cases all items in a coordinated group can be given the same replenishment period. In (Janssen and de Kok [51]) a two-supplier periodic model is discussed where one supplier delivers a fixed quantity while the amount delivered by the other is governed by an (R,S) policy. In (Smits et al. [82]) a production-inventory problem with compound renewal item demand is considered. The model consists of stock-points, one for each item, controlled according to (R,S)-policies and one machine which replenishes them. Periodic review also allows a reasonable prediction of the level of the workload on the staff involved, and is particularly suitable for advanced planning environments and risk management (Tang [85]). For these reasons (R,S) is a popular inventory policy.

As pointed in (Graves [40]) one major theme in the continuing development of inventory theory is to incorporate more realistic assumptions about product demand into inventory models. In most industrial contexts, demand is uncertain and hard to forecast. Many demand histories behave like random walks that evolve over time with frequent changes in their directions and rates of growth or decline. Furthermore, as product life cycles get shorter, the randomness and unpredictability of these demand processes have become even greater. In practice, for such demand processes, inventory managers often rely on forecasts based on a time series of prior demand, such as a weighted moving average. Typically these forecasts are predicated on a belief that the most recent demand observations are the best predictors for future demand. An important class of stochastic production/inventory

control problems therefore assumes a non-stationary demand process. Under this assumption the (R,S) policy takes the non-stationary form (R_n,S_n) where R_n denotes the length of the n^{th} replenishment cycle and S_n the corresponding order-up-to-level (Fig. 6.1). To compute the *near* optimal policy parameters, (Tarim and Kingsman [90]) propose a mixed integer programming (MIP) formulation using a piecewise linear approximation to a complex cost function.

This paper focuses on the work of Tarim and Kingsman, in which a finite-horizon, single-installation, single-item (R_n,S_n) policy is addressed. They assume a fixed procurement cost each time a replenishment order is placed, whatever the size of the order, and a linear holding cost on any unit carried over in inventory from one period to the next. Instead of employing a service level constraint — the probability that at the end of every time period the net inventory will not be negative is at least a certain value (see Bookbinder and Tan [15], Tarim and Kingsman [89] for (R_n,S_n) under a service level constraint) — their model employs a penalty cost scheme. They propose a certainty-equivalent formulation of the above problem in the form of a mixed integer programming (MIP) model. So far no constraint programming (CP) approach has been proposed for (R_n,S_n) under a penalty cost. In fact, as shown in (Tarim and Kingsman [90]), the cost structure is complex in this case and it differs significantly from the one under a service level constraint. (Tarim and Smith [92]) proposed a CP model under a service level constraint. In this paper it was shown that not only CP is able to provide a more compact formulation than the MIP one, but that it is also able to perform faster and to take advantage of dedicated pre-processing techniques that reduce the size of decision variable domains. Moreover dedicated cost-based filtering techniques were proposed in (Tarim et al. [87]) for the same model, these techniques are able to improve performances of several orders of magnitude.

In this paper, we give an *exact* formulation of the (R_n,S_n) inventory control problem via constraint programming, instead of employing a piecewise linear approximation to the total expected cost function. This exact CP formulation provides an optimal solution to (R,S) policy. Our contribution is two-fold: we can now obtain provably optimal solutions, and we can gauge the accuracy of the piecewise linear approximation proposed by Tarim and Kingsman. Furthermore we propose a dedicated *cost-based filtering* method (Focacci and Milano [31]) to

improve performances of the search. The experiments presented show the effectiveness of our approach.

6.2 Problem definition and (R_n, S_n) policy

The demand d_t in period t is considered to be a normally distributed random variable with known probability density function (PDF) $g_t(d_t)$, and is assumed to occur instantaneously at the beginning of each period. The mean rate of demand may vary from period to period. Demands in different time periods are assumed to be independent. A fixed holding cost h is incurred on any unit carried over in inventory from one period to the next. Demands occurring when the system is out of stock are assumed to be back-ordered and satisfied as soon as the next replenishment order arrives. A fixed shortage cost s is incurred for each unit of demand that is back-ordered. A fixed procurement (ordering or set-up) cost a is incurred each time a replenishment order is placed, whatever the size of the order. In addition to the fixed ordering cost, a proportional direct item cost v is incurred. For convenience, and without loss of generality, the initial inventory level is set to zero and the delivery lead-time is not incorporated. It is assumed that negative orders are not allowed, so that if the actual stock exceeds the order-up-to-level for that review, this excess stock is carried forward and does not return to the supply source. However, such occurrences are regarded as rare events and accordingly the cost of carrying the excess stock is ignored. The above assumptions hold for the rest of this paper.

The general multi-period production/inventory problem with stochastic demands can be formulated as finding the timing of the stock reviews and the size of non-negative replenishment orders, X_t in period t, minimizing the expected total cost over a finite planning horizon of N periods:

$$\min \; E\{TC\} = \int_{d_1}\int_{d_2}\dots\int_{d_N}\sum_{t=1}^{N}\left(a\delta_t + vX_t + hI_t^+ + sI_t^-\right) g_1(d_1)\dots g_N(d_N)\mathrm{d}(d_1)\dots\mathrm{d}(d_N) \tag{6.1}$$

subject to

$$X_t > 0 \Rightarrow \delta_t = 1 \tag{6.2}$$

$$I_t = \sum_{i=1}^{t} (X_i - d_i) \tag{6.3}$$

$$I_t^+ = \max(0, I_t) \tag{6.4}$$

$$I_t^- = -\min(0, I_t) \tag{6.5}$$

$$X_t, I_t^+, I_t^- \in \mathbb{Z}^+ \cup \{0\}, \quad I_t \in \mathbb{Z}, \quad \delta_t \in \{0, 1\} \tag{6.6}$$

for $t = 1 \ldots N$, where

d_t : the demand in period t, a normal random variable with PDF $g_t(d_t)$,
a : the fixed ordering cost,
v : the proportional direct item cost,
h : the proportional stock holding cost,
s : the proportional shortage cost,
δ_t : a {0,1} variable that takes the value of 1 if a replenishment occurs in period t and 0 otherwise,
I_t : the inventory level at the end of period t, $-\infty < I_t < +\infty$, $I_0 = 0$
I_t^+ : the excess inventory at the end of period t carried over to the next period, $0 \leq I_t^+$,
I_t^- : the shortages at the end of period t, or magnitude of negative inventory $0 \leq I_t^-$,
X_t : the replenishment order placed and received in period t, $X_t \geq 0$.

The proposed non-stationary (R,S) policy consists of a series of review times and associated order-up-to-levels. Consider a review schedule which has m reviews over the N period planning horizon with orders arriving at $\{T_1, T_2, \ldots, T_m\}$, $T_j > T_{j-1}$. For convenience $T_1 = 1$ is defined as the start of the planning horizon and $T_{m+1} = N + 1$ the period immediately after the end of the horizon. In (Tarim and Kingsman [90]), the decision variable X_{T_i} is expressed in terms of a new variable $S_t \in \mathbb{Z}$, where S_t may be interpreted as the opening stock level for period t, if there is no replenishment in this period (i.e. $t \neq T_i$ and $X_t = 0$) and the order-up-to-level for the i-th review period T_i if there is a replenishment (i.e.

$t = T_i$ and $X_t > 0$). According to this transformation the expected cost function, Eq. (6.1), is written as the summation of m intervals, T_i to T_{i+1} for $i = 1, \ldots, m$, defining $D_{t_1,t_2} = \sum_{j=t_1}^{t_2} d_j$:

$$\begin{aligned} \min \ E\{TC\} = & \sum_{i=1}^{m} \left(a\delta_{T_i} + \sum_{t=T_i}^{T_{i+1}-1} E\{C_{T_i,t}\} \right) + \\ & vI_N + v \int_{D_{1,N}} D_{1,N} \times g(D_{1,N}) \mathrm{d}(D_{1,N}), \end{aligned} \tag{6.7}$$

The term $v \int_{D_{1,N}} D_{1,N} \times g(D_{1,N}) \mathrm{d}(D_{1,N})$ is constant and can therefore be ignored in the optimization model. $E\{C_{T_i,t}\}$ of Eq. (6.7) is defined as:

$$\int_{-\infty}^{S_{T_i}} h \left(S_{T_i} - D_{T_i,t}\right) g(D_{T_i,t}) \mathrm{d}(D_{T_i,t}) - \int_{S_{T_i}}^{\infty} s \left(S_{T_i} - D_{T_i,t}\right) g(D_{T_i,t}) \mathrm{d}(D_{T_i,t}). \tag{6.8}$$

As stated in (Tarim and Kingsman [90]), $E\{C_{T_i,t}\}$ is the expected cost function of a single-period inventory problem where the single-period demand is $D_{T_i,t}$. Since S_{T_i} may be interpreted as the order-up-to-level for the i-th review period T_i and $S_{T_i} - D_{T_i,t}$ is the end of period inventory for the "single-period" with demand $D_{T_i,t}$, the expected total subcosts $E\{C_{T_i,t}\}$ are the sums of single-period inventory costs where the demands are the cumulative demands over increasing periods.

By dropping the T_i and t subscripts in Eq. (6.8) we obtain the following well-known expression for the expected total cost of a single-period newsvendor problem:

$$E\{TC\} = h \int_{-\infty}^{S} (S - D) g(D) \mathrm{d}(D) - s \int_{S}^{\infty} (S - D) g(D) \mathrm{d}(D) \tag{6.9}$$

where we consider two cost components: holding cost on the positive end of period inventory and shortage cost for any back-ordered demand. Let $G(\cdot)$ be the cumulative distribution function of the demand in our single-period newsvendor problem. A known result in inventory theory (Hadley and Whitin [41]) is convexity of Eq. (6.9). The so-called *Critical Ratio*, $\frac{s}{s+h}$, can be seen as the service level β (i.e. probability that at the end of the period the inventory level is non-

negative) provided when we fix the order-up-to-level S to the optimal value S^* that minimizes expected holding and shortage costs (Eq. (6.9)). By assuming $G(\cdot)$ to be strictly increasing, we can compute the optimal order-up-to-level as $S^* = G^{-1}\left(\frac{s}{s+h}\right)$.

6.2.1 Stochastic cost component in single-period newsvendor

We now aim to characterize the cost of the policy that orders S^* units to meet the demand in our single-period newsvendor problem. Such a problem has been widely studied in the inventory control literature (Silver et al. [81]). Since the demand D is assumed to be normal with mean μ and standard deviation σ, then we can write $D = \mu + \sigma Z$, where Z is a standard normal random variable. Let $\Phi(z) = \Pr(Z \leq z)$ be the cumulative distribution function of the standard normal random variable. Since $\Phi(\cdot)$ is strictly increasing, $\Phi^{-1}(\cdot)$ is uniquely defined. Let $z_\beta = \Phi^{-1}(\beta)$, since $\Pr(D \leq \mu + z_\beta\sigma) = \Phi(z_\beta) = \beta$, it follows that $S^* = \mu + z_\beta\sigma$. The quantity z_β is known as the safety factor and $S^* - \mu = z_\beta\sigma$ is known as the safety stock. It can be shown (Hadley and Whitin [41]) that

$$\int_{S^*}^{\infty}(S^*-D)g(D)\mathrm{d}(D) = E\{D-S^*\}^+ = \sigma E\{Z-z_\beta\}^+ = \sigma[\phi(z_\beta)-(1-\beta)z_\beta] \tag{6.10}$$

where $\phi(\cdot)$ is the PDF of the standard normal random variable. Let $E\{S^*-D\}^+ = \int_{-\infty}^{S}(S-D)g(D)\mathrm{d}(D)$, it follows

$$\begin{aligned} E\{TC(S^*)\} = h\cdot E\{S^*-D\}^+ + s\cdot E\{D-S^*\}^+ = \\ h\cdot(S^*-\mu) + (h+s)E\{D-S^*\}^+ = \\ hz_\beta\sigma + (h+s)\sigma E\{Z-z_\beta\}^+ = \\ hz_\beta\sigma + (h+s)\sigma[\phi(z_\beta) - (1-\beta)z_\beta] = \\ (h+s)\sigma\phi(z_\beta) \end{aligned} \tag{6.11}$$

The last expression $(h+s)\sigma\phi(z_\beta)$ holds only for the optimal order-up-to-level S^* that provides the service level $\beta = \left(\frac{s}{s+h}\right)$ computed from the *critical ratio* (CR).

Instead, expression

$$hz_\alpha\sigma + (h+s)\sigma[\phi(z_\alpha) - (1-\alpha)z_\alpha] \tag{6.12}$$

can be used to compute the expected total cost for any given level S such that $\alpha = \Phi\left(\frac{S-\mu}{\sigma}\right)$. In Fig. 6.2 we plot this cost for a particular instance as a function of the opening inventory level S.

6.2.2 Stochastic cost component in multiple-period newsvendor

The considerations in the former sections refer to a single-period problem, but they can be easily extended to a replenishment cycle $R(i,j)$ that covers the period span $i,\ldots,j$. In (Levi et al. [59]) it is possible to find a discussion on multi-period newsvendor problems and a sampling-based heuristic approach to find near-optimal solutions. In contrast the approach we propose is exact. The demand in each period is normally distributed with PDF $g_i(d_j),\ldots,g_j(d_j)$. The cost for the multiple periods' replenishment cycle, when ordering costs are neglected, can be expressed as

$$E\{TC\} = \sum_{k=i}^{j}\left(h\int_{-\infty}^{S}(S-d_{i,k})g_{i,k}(d_{i,k})\mathrm{d}(d_{i,k}) - s\int_{S}^{\infty}(S-d_{i,k})g_{i,k}(d_{i,k})\mathrm{d}(d_{i,k})\right) \tag{6.13}$$

Since demands are independent and normally distributed in each period, the term $g_{i,j}(d_{i,j})$ (that is the p.d.f. for the overall demand over the period span $\{i,\ldots,j\}$) can be easily computed (Fortuin [34]) once the demand in each period $d_i,\ldots,d_j$ are known. It is easy to apply the same rule as before and compute the second derivative of this expression:

$$\frac{\mathrm{d}^2}{\mathrm{d}S^2}E\{TC\} = \sum_{k=i}^{j}\left(h\cdot g_{i,k}(S) + s\cdot g_{i,k}(S)\right) \tag{6.14}$$

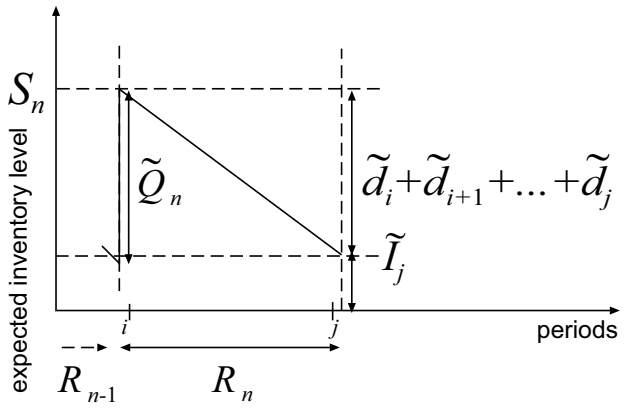


Figure 6.1: (R_n,S_n) policy. $\tilde{d}_i + \tilde{d}_{i+1} + \ldots + \tilde{d}_j$ is the expected demand over R_n; $\tilde{I}_j = S_n - \tilde{d}_i + \tilde{d}_{i+1} + \ldots + \tilde{d}_j$ is the expected closing inventory level for R^n.

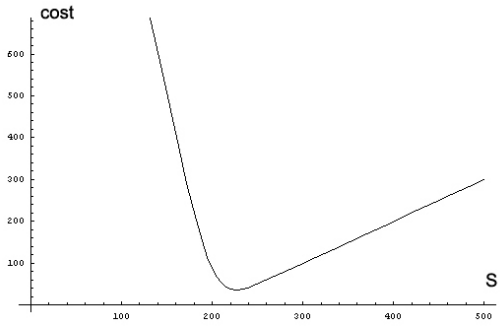


Figure 6.2: Single-period holding and shortage cost as a function of the opening inventory level S. The demand is normally distributed with mean 200 and standard deviation 20. Holding cost is 1, shortage cost is 10.

which is again a positive function of S, since $g_{i,k}(S)$ are PDFs and both holding and shortage cost are assumed to be positive. The expected cost of a single replenishment cycle therefore remains convex in S regardless of the periods covered. Unfortunately it is not possible to compute the CR as before, using a simple algebraic expression to obtain the optimal S^* which minimizes the expected cost. But since the cost function is convex, it is still possible to compute S^* efficiently. Eq. (6.12) can be extended in the following way to compute the cost for the replenishment cycle $R(i,j)$ as a function of the opening inventory level S:

$$\sum_{k=i}^{j} \left(h z_{\alpha(i,k)} \sigma_{i,k} + (h+s)\sigma_{i,k}[\phi(z_{\alpha(i,k)}) - (1-\alpha(i,k)) z_{\alpha(i,k)}] \right) \tag{6.15}$$

where $G_{i,k}(S) = \alpha(i,k)$ and $z_{\alpha(i,k)} = \Phi^{-1}(\alpha(i,k))$. Therefore we have $j-i+1$ cost components: the holding and shortage cost at the end of period $i, i+1, \ldots, j$. In Fig. 6.3 we plot this cost for a particular instance as a function of the opening inventory level S. For each possible replenishment cycle we can efficiently compute

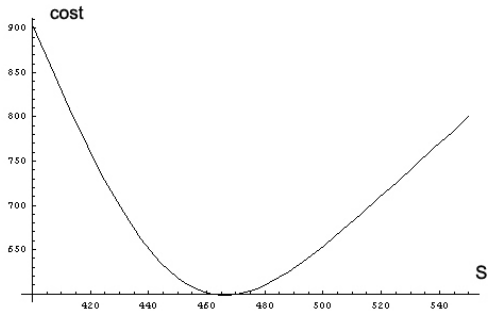


Figure 6.3: Three periods holding and shortage cost as a function of the opening inventory level S. The demand is normally distributed in each period with mean respectively 150, 100, 200, the coefficient of variation is 0.1. Holding cost is 1, shortage cost is 10.

the optimal S^* that minimizes such a cost function, using gradient based methods for convex optimization such as Newton's method. Notice that the complete expression for the cost of replenishment cycles that start in period $i \in \{1, ..., N\}$

and end in period N is

$$\sum_{k=i}^{N} \left(h z_{\alpha(i,k)} \sigma_{i,k} + (h+s)\sigma_{i,k}[\phi(z_{\alpha(i,k)}) - (1-\alpha(i,k))z_{\alpha(i,k)}] \right) + v\left(S - \sum_{k=i}^{N} d_k \right) \tag{6.16}$$

In fact for this set of replenishment cycles we must also consider the unit cost component. Once S^* is known, by subtracting the expected demand over the replenishment cycle we obtain the optimal expected buffer stock level $b(i,j)$ required for such a replenishment cycle in order to minimize holding and shortage cost. Notice that every other choice for buffer stock level will produce a higher expected total cost for $R(i,j)$.

6.2.3 Upper-bound for opening inventory levels

We now propose an *upper bound* for the value of the opening inventory level in each period $t \in \{1,...,N\}$. Firstly we ignore the direct item cost v, in fact from Eq. (6.7) it is trivial to see that v may only decrease the opening inventory level for the last replenishment cycle scheduled. We consider a single replenishment cycle covering the whole planning horizon. If we relax the original problem formulation and we ignore holding and shortage cost components at the end of each period $t \in \{1,...,N-1\}$, the resulting model will reflect a single period newsvendor problem. In this problem we incur holding and shortage cost only at the end of the last period N and the stochastic demand is given by the sum of the demand distributions in each period of our planning horizon. The optimal buffer stock $b(1,N)$ required to optimize the convex cost for this problem can be easily computed, as seen, by means of the critical ratio. It is easy to see that, since we relaxed holding and shortage costs for each period $t \in \{1,...,N-1\}$, then for each period $t \in \{1,...,N\}$, $\max(S_t) = \sum_t^N \tilde{d}_t$+$b(1,N)$. In fact, since we assume a shortage cost higher than holding cost, opening inventory levels for this replenishment cycle may only be decreased by the additional cost components in the original model. Moreover the upper bounds computed are still valid if the plan-

ning horizon is covered by more than a single replenishment cycle. The reason is the following. If the planning horizon is covered by a number of replenishment cycles, again it is possible to apply a similar reasoning and it is possible to reduce each replenishment cycle R_k covering periods $\{i, \ldots, j\}$ to a single period newsvendor problem, by ignoring holding and shortage costs for each period $t \in \{i, \ldots, j-1\}$ and by considering only the cost component of the last period j. Then for each replenishment cycle R_k we will easily obtain a buffer stock $b(i, j)$, by means of the critical ratio. Since $b(i, j)$ is increasing, that is $b(i, j) \leq b(i, j+1)$, as shown in (Tarim and Smith [92]), obviously opening inventory levels computed in this case will be lower than those computed for the former case where a single replenishment cycle covers the whole planning horizon. Furthermore we recall that also in this case opening inventory levels may only be decreased when the additional holding and shortage cost components for other periods are reintroduced in the model. It directly follows that the upper bounds computed are valid for the original model.

6.2.4 Lower-bound for expected closing inventory levels

A *lower bound* for the value of the expected closing inventory level in each period $t \in \{1, ..., N\}$, that is opening inventory level minus expected demand, can be computed by considering every possible buffer stock $b(i, j)$ required to optimize the convex cost of a single replenishment cycle $R(i, j)$, independently of the other cycles that are planned. The lower bound will be the minimum value among all these possible buffer values for $j \in \{1, ..., N\}$ and $i \in \{1, ..., j\}$.

6.3 Deterministic equivalent CP formulation

Building on the considerations above it is easy to construct a *deterministic equivalent* CP formulation for the non-stationary (R_n, S_n) policy under stochastic demand, ordering cost, holding and shortage cost. (For a detailed discussion on deterministic equivalent modeling in stochastic programming see Birge and Louveaux [11]).

In order to correctly compute the expected total cost for a replenishment cycle

$R(i,j)$ with opening inventory level S_i, we must build a special-purpose constraint $objConstraint(\cdot)$ that dynamically computes such a cost by means of an extended version of Eq. (6.15)

$$C(S_i,i,j) = a + \sum_{k=i}^{j} \left(h z_{\alpha(i,k)} \sigma_{i,k} + (h+s)\sigma_{i,k}[\phi(z_{\alpha(i,k)}) - (1-\alpha(i,k)) z_{\alpha(i,k)}] \right) \tag{6.17}$$

that considers the ordering cost. Then the expected total cost for a certain replenishment plan will be computed as the sum of all the expected total costs for replenishment cycles in the solution, plus the respective ordering costs. $objConstraint(\cdot)$ also computes the optimal expected buffer stock level $b(i,j)$ for every replenishment cycle $R(i,j)$ identified by a partial assignment for $\delta_{k\in\{1,\ldots,N\}}$ variables. A *deterministic equivalent* CP formulation is

$$\min \;\; E\{TC\} = C \tag{6.18}$$

subject to

$$\text{objConstraint}\left(C, \tilde{I}_1, \ldots, \tilde{I}_N, \delta_1, \ldots, \delta_N, d_1, \ldots, d_N, a, h, s\right) \tag{6.19}$$

and for $t = 1 \ldots N$

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} \geq 0 \tag{6.20}$$

$$\tilde{I}_t + \tilde{d}_t - \tilde{I}_{t-1} > 0 \Rightarrow \delta_t = 1 \tag{6.21}$$

$$\tilde{I}_t \in \mathbf{Z}, \quad \delta_t \in \{0,1\} \tag{6.22}$$

Each decision variable $\tilde{I}_t$ represents the expected closing inventory level at the end of period t; bounds for the domains of these variables can be computed as explained above. Each $\tilde{d}_t$ represents the expected value of the demand in a given period t according to its PDF $g_t(d_t)$. The binary decision variables δ_t state whether a replenishment is fixed for period t ($\delta_t = 1$) or not ($\delta_t = 0$).

Eq. (6.20) enforces a no-buy-back condition, which means that received goods cannot be returned to the supplier. As a consequence of this the expected inventory level at the end of period t must be no less than the expected inventory level at the

end of period $t-1$ minus the expected demand in period t. Eq. (6.21) expresses the replenishment condition. We have a replenishment if the expected inventory level at the end of period t is greater than the expected inventory level at the end of period $t-1$ minus the expected demand in period t. This means that we received some extra goods as a consequence of an order.

The objective function (6.18) minimizes the expected total cost over the given planning horizon. $objConstraint(\cdot)$ dynamically computes buffer stocks and it assigns to C the expected total cost related to a given assignment for replenishment decisions, depending on the demand distribution in each period and on the given combination for problem parameters a, h, s. In order to propagate this constraint we wait for a partial assignment involving δ_t, $t = 1, \ldots, N$ variables. In particular we look for an assignment where there exists some i s.t. $\delta_i = 1$, some $j > i$ s.t. $\delta_{j+1} = 1$ and for every k, $i < k \leq j$, $\delta_k = 0$. This will uniquely identify a replenishment cycle $R(i, j)$ (Fig. 6.4). There may be more replenishment

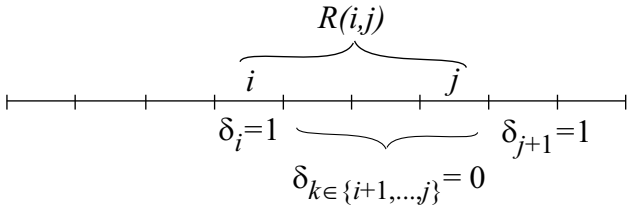


Figure 6.4: A replenishment cycle $R(i, j)$ is identified by the current partial assignment for δ_i variables.

cycles associated with a partial assignment. If we consider each $R(i, j)$ identified by the current assignment, it is easy to minimize the convex cost function already discussed, and to find the optimal expected buffer stock $b(i, j)$ for this particular replenishment cycle independently on the others. By doing this for every replenishment cycle identified, two possible situations may arise: the buffer stock configuration obtained satisfies every inventory conservation constraint (Eq. (6.20)), or for some couple of subsequent replenishment cycles this constraint is violated (Fig. 6.5). Therefore we observe an expected negative order quantity. If the latter situation arises we can adopt a fast convex optimization procedure to compute a feasible buffer stock configuration with minimum cost. The key idea is to identify two possible limit situations: we increase the opening inventory level of the

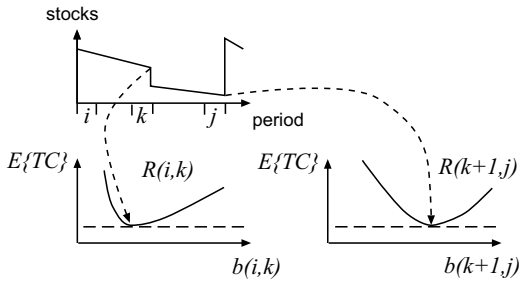


Figure 6.5: The expected total cost of both replenishment cycles is minimized, but the inventory conservation constraint is violated between $R(i,k)$ and $R(k+1,j)$

second cycle, thus incurring a higher overall cost for it, to preserve optimality of the first cycle (Fig. 6.6 - a). Or we decrease the buffer stock of the first replenishment cycle, thus incurring a higher overall cost for it, to preserve optimality of the second cycle cost (Fig. 6.6 - b). A key observation is that, when negative

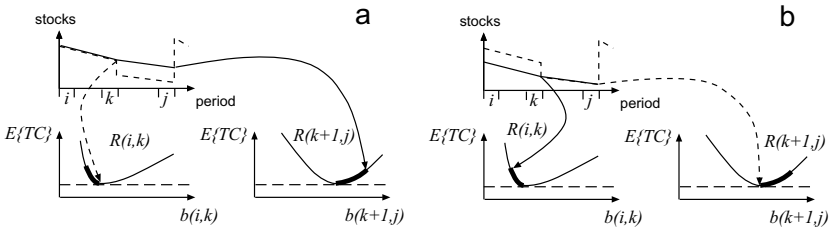


Figure 6.6: Feasible limit situations when negative order quantity scenarios arise

order quantity scenarios arise, at optimality the expected closing inventory levels of the first and the second cycle lie in the interval delimited by the two situations described. This directly follows from the convexity of both the cost functions. Moreover the expected closing inventory level of the first cycle must be equal to the opening inventory level of the second cycle. In fact, if this does not hold, then either the first cycle has an expected closing inventory level higher than the opening inventory level of the second cycle and the solution is not feasible (Fig. 6.7 - a), or the first cycle has an expected closing inventory level smaller than the opening inventory level of the second cycle. In the latter case we can obviously decrease the overall cost by choosing a smaller opening inventory level for the second cycle (Fig. 6.7 - b). The algorithm for computing optimal buffer stock configurations in presence of negative order quantity scenarios simply exploits

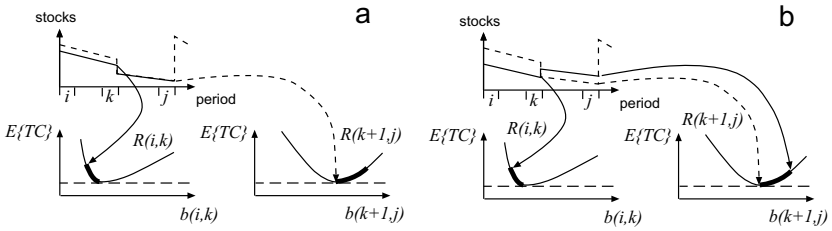


Figure 6.7: Infeasible (a) and suboptimal (b) plans realized when the opening inventory level of the second cycle doesn't equate the expected closing inventory level of the first cycle

the linear dependency between the opening inventory level of the second cycle and the expected closing inventory level of the first cycle. Due to this dependency the overall cost is still convex in $b(i,k)$ (or equivalently in $b(k+1,j)$, since they are linearly dependent) and we can apply any convex optimization technique to find the optimal buffer stock configuration. Notice that this reasoning still holds in a recursive process. Therefore we can optimize buffer stock for two subsequent replenishment cycles, then we can treat these as a new single replenishment cycle, since their buffer stocks are linearly dependent, and repeat the process in order to consider the next replenishment cycle if a negative order quantity scenario arises.

Once buffer stocks are known we can apply Eq. (6.17) to the opening inventory level $S_i = \tilde{d}_i + \ldots + \tilde{d}_j + b(i,j)$ and compute the cost $C(S_i, i, j)$ associated with a given replenishment cycle. Since the cost function in Eq. (6.17) is convex and we handle negative order quantity scenarios, a lower bound for the expected total cost associated with the current partial assignment for δ_t, $t = 1, \ldots, N$ variables is now given by the sum of all the cost components $C(S_i, i, j)$, for each replenishment cycle $R(i,j)$ identified by the assignment. Furthermore this bound is tight if all the δ_t variables have been assigned. $objConstraint(\cdot)$ exploits this property in order to incrementally compute a lower bound for the cost of the current partial assignment for δ_t variables. When every δ_t variable is ground, since such a lower bound becomes tight, buffer stocks computed for each replenishment cycle identified can be assigned to the respective I_t variables. Finally, in order to consider the unit variable cost v we must add the term $v \cdot I_N$ to the cycle cost $C(S_i, i, N)$ for $i \in \{1, ..., N\}$. Therefore the complete expression for the cost of

Period	1	2	3	4	5	6	7	8
$\tilde{d}_t$	200	100	70	200	300	120	50	100

Table 6.1: Expected demand values

replenishment cycles that start in period $i \in \{1, ..., N\}$ and end in period N is:

$$C(S_i, i, N) = a + \sum_{k=i}^{N} \left(h z_{\alpha(i,k)} \sigma_{i,k} + (h+s)\sigma_{i,k}[\phi(z_{\alpha(i,k)}) - (1-\alpha(i,k)) z_{\alpha(i,k)}] \right) + v \left(S_i - \sum_{k=i}^{N} d_k \right) \tag{6.23}$$

6.4 Comparison of the CP and MIP approaches

(Tarim and Kingsman [90]) proposed a piecewise linear approximation of the cost function for the single-period newsvendor type model under holding and shortage costs, which we analyzed above. Thus they were able to build a MIP model approximating an optimal solution for the multi-period stochastic lot-sizing under fixed ordering, holding and shortage costs. They gave a few examples to show the effect of higher noise levels (uncertainty in the demand forecasts) on the order schedule. Using the same examples we shall compare the policies obtained using our exact CP approach with their approximation. Depending on the number of segments used in the piecewise approximation, the quality of the solutions obtained can be improved. We shall consider approximations with two and seven segments. The forecast of demand in each period are given in Table 6.1. We assume that the demand in each period is normally distributed about the forecast value with the same coefficient of variation τ. Thus the standard deviation of demand in period t is $\sigma_t = \tau \cdot \tilde{d}_t$. In all cases, initial inventory levels, delivery lead-times and salvage values are set to zero.

In Fig. 6.8–6.12 optimal replenishment policies obtained with our CP approach are compared for four different instances, with respect to τ, v, a and s, with the policies provided by the 2-segment (PW-2) and 7-segment (PW-7) approximations. For each instance we compare the expected total cost provided by the exact

method with the expected total cost provided by the policies found using approximate MIP models. Since the cost provided by PW-2 and PW-7 is an approximation, it often differs significantly from the real expected total cost related to policy parameters found by these models. It is therefore not meaningful to compare the cost provided by the MIP model with that of the optimal policy obtained with our CP model. To obtain a meaningful comparison we computed the real expected total cost by applying the exact cost function (Eqs. 6.17, 6.23) discussed above to the (R_n,S_n) policy parameters obtained through PW-2 and PW-7. It is then possible to assess the accuracy of approximations in (Tarim and Kingsman [90]). Fig.

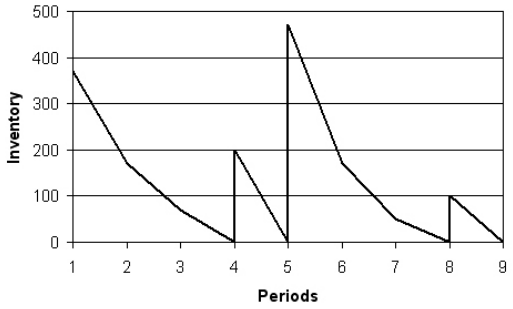


Figure 6.8: $h = 1$, $a = 250$, $s = 10$, $v = 0$, $\tau = 0.0$

6.8 shows the optimal replenishment policy for the deterministic case ($\tau = 0.0$). The direct item cost (v) is taken as zero. Four replenishment cycles are planned. The (R_n,S_n) policy parameters are $R = [3, 1, 3, 1]$ and $S = [370, 200, 470, 100]$. The total cost for this policy is 1460. Fig. 6.9 shows an instance where we con-

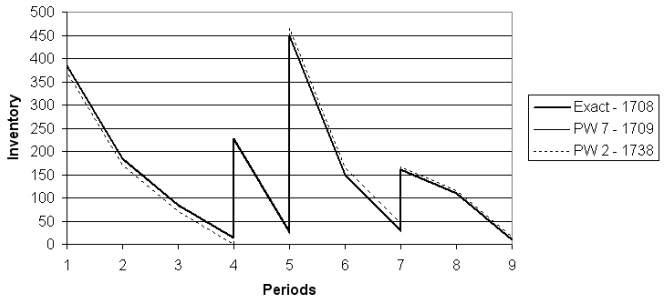


Figure 6.9: $h = 1$, $a = 250$, $s = 10$, $v = 0$, $\tau = 0.1$

sider low levels of forecast uncertainty ($\tau = 0.1$). In this case both PW-2 and

PW-7 perform well compared to our exact CP solutions. Since forecast uncertainty must be considered, all the models introduce buffer stocks. The optimal (R_n,S_n) policy parameters found by our CP approach are $R = [3, 1, 2, 2]$ and $S = [384, 227, 449, 160]$. The PW-2 solution is 1.75% more costly than the exact solution, while the PW-7 solution is slightly more costly than the exact solution. Fig. 6.10 shows that as the level of forecast uncertainty increases ($\tau = 0.2$), the

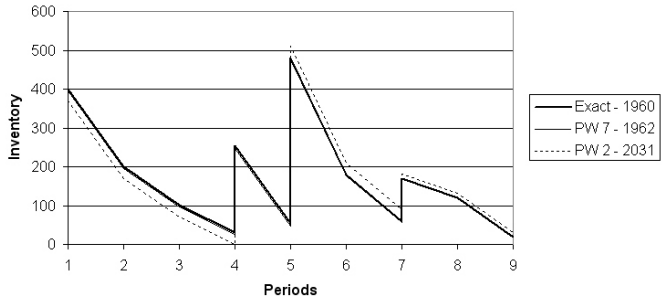


Figure 6.10: $h = 1$, $a = 250$, $s = 10$, $v = 0$, $\tau = 0.2$

quality of the PW-2 solution deteriorates, in fact it is now 3.62% more costly than the exact solution. The optimal (R_n,S_n) policy parameters found by our CP approach are $R = [3, 1, 2, 2]$ and $S = [401, 253, 479, 170]$. In contrast the PW-7 solution is still only slightly more costly than the exact solution. As noted in

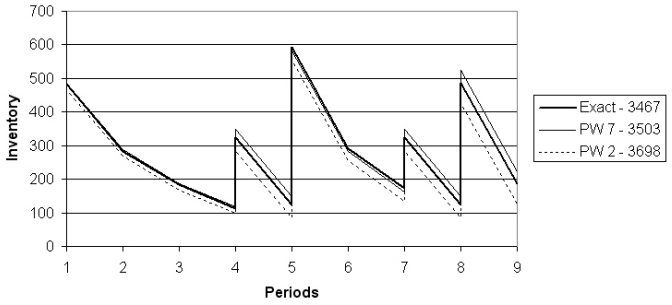


Figure 6.11: $h = 1$, $a = 350$, $s = 50$, $v = 0$, $\tau = 0.3$

(Tarim and Kingsman [90]) the quality of the approximation decreases for high ratios s/h. In Fig. 6.11 we consider $s/h = 50$ and a different demand pattern. The forecast of demand in each period are given in Table 6.2. Now the PW-2 solution is 6.66% more costly than the exact approach, while the PW-7 solution

Period	1	2	3	4	5	6	7	8
$\bar{d}_t$	200	100	70	200	300	120	200	300

Table 6.2: Expected demand values

is 1.03% more costly. The optimal (R_n,S_n) policy parameters found by our CP approach are $R = [3, 1, 2, 1, 1]$ and $S = [483, 324, 592, 324, 486]$. In Fig. 6.12 we consider the same instance but a direct item cost is now incurred ($v = 15$). The buffer stock held in the last replenishment cycle is affected by this parameter, and is decreased from 186 to 63. The PW-7 policy is now 0.84% more costly than the exact one. For these instances seven segments usually provides a solution with a

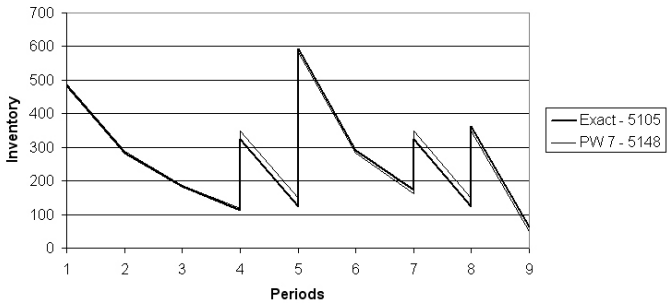


Figure 6.12: $h = 1$, $a = 350$, $s = 50$, $v = 15$, $\tau = 0.3$

cost reasonably close to optimal. In terms of running times, for all these instances both the MIP approximations and the CP model perform very quickly. In our experiments we used ILOG OPL Studio 3.7 to solve the MIP models of (Tarim and Kingsman [90]), and Choco ([58] an open source solver written in Java) to implement our CP model. All experiments were performed on an Intel Centrino 1.5 GHz with 500Mb RAM. Since the planning horizon is short (8 periods), we were able to solve any instance in less than a second. As the planning horizon length increases the pure CP model becomes slower than the MIP one. This is due both to the size of decision variable domains and to the lack of good bounds in the search.

In the following sections we will discuss how it is possible to incorporate in our CP model a dedicated cost-based filtering method (Focacci and Milano [31]) based on a *dynamic programming relaxation* (Tarim [86]) that is able to generate

good bounds during the search. Such a technique has been already employed under a service level constraint (Tarim et al. [87]). It should be noted that due to the non-linearity of the cost function induced by the shortage cost scheme, the version of the problem we consider is significantly more complicated than the one under a service level constraint. Nevertheless, despite the non-linearity of the cost function, we will see that the convexity of the cost function can be exploited to define a relaxation similar to the one proposed in (Tarim et al. [87]).

6.4.1 Cost-based filtering by relaxation

Cost-based filtering is an elegant way of combining techniques from CP and Operations Research (OR) (Fahle and Sellmann [28], Focacci and Milano [31]). OR-based optimization techniques are used to remove values from variable domains that cannot lead to better solutions. This type of domain filtering can be combined with the usual CP-based filtering methods and branching heuristics, yielding powerful hybrid search algorithms.

In (Tarim et al. [87]) the authors adopt a relaxation proposed by (Tarim [86]) for the CP model that computes (R_n,S_n) policy parameters under service level constraints. When the relaxed model is solved it provides good bounds for the original problem. Furthermore the relaxed problem is a Shortest Path Problem that can be solved in polynomial time. Therefore it is easy to obtain good bounds at each node of the search tree. In the same work the authors also explain how it is possible to take into account a partial assignment for replenishment decisions $\delta_1, \ldots, \delta_N$ and for expected closing inventory levels $\tilde{I}_1, \ldots, \tilde{I}_N$ when the relaxed problem is constructed, so that the effect of these assignments is reflected on the bound that is obtained by solving the relaxed problem. As shown in (Tarim et al. [87]), the CP model proposed for computing (R_n,S_n) policy parameters under service level constraints can be reduced to a Shortest Path Problem if the inventory conservation constraint and the replenishment condition constraint, that is constraint 6.20 and 6.21 in our model under shortage cost scheme, are relaxed for replenishment periods. That is for each possible pair of replenishment cycles $\langle R(i, k-1), R(k, j)\rangle$ where $i, j, k \in \{1, \ldots, N\}$ and $i < k \leq j$, the relationship between the opening inventory level of $R(k, j)$ and the expect closing inventory

level of $R(i, k-1)$ is not considered. The same approach can be translated to the CP model for (R_n,S_n) under shortage cost scheme.

In the former sections we provided a general function $C(i, j, \tilde{I}_j)$ to compute the expected total cost of replenishment cycle $R(i, j)$, when an expected closing inventory level $\tilde{I}_j$ is held in period j. Furthermore we proved that this function is convex in $\tilde{I}_j$ (Fig. 6.13). If we consider each replenishment cycle $R(i, j)$ independently, we can efficiently compute the optimal expected closing inventory level that minimizes the expected total cost associated with such a cycle using gradient based methods for convex optimization. This way we obtain a set $\mathcal{S}$ of $N(N+1)/2$ possible replenishment cycles and respective order-up-to-levels. Our new problem is to find an optimal set $\mathcal{S}^* \subset \mathcal{S}$ of consecutive disjoint replenishment cycles that covers our planning horizon at the minimum cost. In (Tarim et al. [87]) it was shown that the optimal solution to this relaxation is given by the shortest path in a graph from a given initial node to a final node where each arc represents a specific cost. We now adapt their approach to our model that employs a shortage cost scheme.

If N is the number of periods in the planning horizon of the original problem, we introduce $N+1$ nodes. Since we assume, without loss of generality, that an order is always placed at period 1, we take node 1, which represents the beginning of the planning horizon, as the initial one. Node $N+1$ represents the end of the planning horizon. For each possible replenishment cycle $R(i, j-1)$ such that $i, j \in \{1, \ldots, N+1\}$ and $i < j$, we introduce an arc (i, j) with associated cost

$$Q(i, j) = C(i, j-1, \tilde{I}^*_{j-1}), \tag{24}$$

where $\tilde{I}^*_{j-1}$ is the expected closing inventory level that minimizes the convex cost of replenishment cycle $R(i, j-1)$. Since we are dealing with a one-way temporal feasibility problem (Wagner and Whitin [96]), when $i \geq j$, we introduce no arc. The connection matrix for such a graph, of size $N \times (N+1)$, can be built as shown in Table 6.3.

The cost of the shortest path from node 1 to node $N+1$ in the given graph is a valid lower bound for the original problem, as it is a solution of the relaxed problem. In fact the expected total cost function for each replenishment cycle is

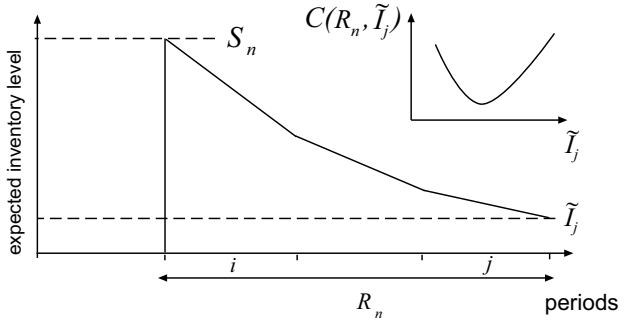


Figure 6.13: Convexity of the expected total cost associated with a given replenishment cycle R_n covering periods $\{i, \ldots, j\}$. The expected total cost is a function of the expected closing inventory level $\tilde{I}_j$.

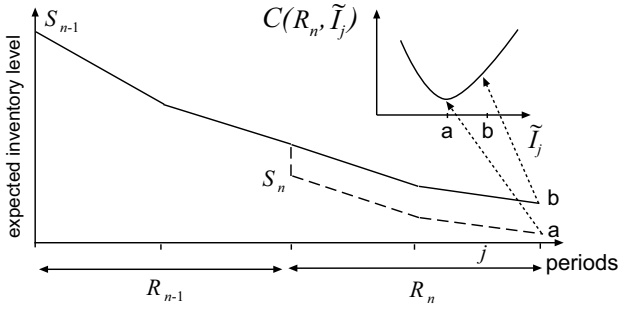


Figure 6.14: The optimal expected closing inventory level for replenishment cycle R_n considered alone is a, this minimizes the convex cost associated with replenishment cycle R_n. In order to meet the inventory conservation constraint for the stocks carried over from cycle R_{n-1}, the minimum expected closing inventory level required is b. Such a value produces a higher expected total cost for R_n.

	1	2	...	j	...	$N+1$
1	–	$Q(1,2)$	...	$Q(1,j)$	...	$Q(1,N+1)$
$\vdots$	–	–	$\ddots$	$\vdots$	$\ddots$	$\vdots$
i	–	–	–	$Q(i,j)$	...	$Q(i,N+1)$
$\vdots$	–	–	–	–	$\ddots$	$\vdots$
N	–	–	–	–	–	$Q(N,N+1)$

Table 6.3: Shortest Path Problem Connection matrix

convex in the expected closing inventory level held at the end of the cycle. Therefore in order to meet the violated inventory conservation constraints, if any exists, we will incur an overall higher expected total cost for a given group of replenishment cycles (Fig. 6.14). Furthermore it is easy to map the optimal solution for the relaxed problem, that is the set of arcs participating to the shortest path, to a solution for the original problem by noting that each arc (i, j) represents a replenishment cycle $R(i, j-1)$. The feasibility of such a solution with respect to the original problem can be checked by verifying that it satisfies every relaxed constraint. If no inventory conservation constraint is violated, it is easy to see that the computed cost is optimal for the given replenishment plan.

We will now show how to exploit this *lower bound* in an *optimization oriented* global constraint able to dynamically produce good bounds when a partial solution is provided. A detailed discussion on optimization oriented global constraints can be found in (Focacci and Milano [33]).

Cost-based filtering can be performed by simply noticing that the costs stored in the connection matrix can be adjusted to reflect the current partial assignment for decision variables δ_t and $\tilde{I}_t$ exactly the way shown for the service level constrained model (Tarim et al. [87]). More specifically:

$\delta_k = 0$**:** If in a given partial solution a decision variable δ_k, $k \in \{1, \ldots, N\}$ has been already set to 0, then we can remove from the graph every inbound arc to node k and every outbound arc from node k. This prevents node k from being part of the shortest path, and hence prevents period k from being a replenishment period. In this modified graph, the cost of the shortest path will provide a valid

lower bound for the cost of an optimal solution incorporating the decision $\delta_k = 0$. Furthermore, an assignment for decision variables is associated with the shortest path. If this assignment is feasible for the original problem, then it is optimal with the respect to the decision $\delta_k = 0$.

$\delta_k = 1$: On the other hand, if in a given partial solution a decision variable δ_k, $k \in \{1, \ldots, N\}$ has been already set to 1, then we can remove from the graph every arc connecting a node i to a node j, where $i < k < j$. This forces the shortest path to pass through node k, and hence forces period k to be a replenishment period. In this modified graph, the cost of the shortest path will provide a valid lower bound for the cost of an optimal solution incorporating the decision $\delta_k = 1$. Furthermore, an assignment for decision variables is associated with the shortest path. If this assignment is feasible for the original problem, then it is optimal with the respect to the decision $\delta_k = 1$.

$\tilde{I}_t$ assigned: If a given $\tilde{I}_t$, $t \in \{i, \ldots, j-1\}$, is assigned a value, the expected closing inventory level ($\tilde{I}_{j-1}$) for the replenishment cycle $R(i, j-1)$, which covers period t, is uniquely determined and therefore the expected total cost for such a replenishment cycle — that is the cost of arc (i, j) — can be directly computed from $C(i, j-1, \tilde{I}_{j-1})$, provided that the current partial assignment for δ_t decision variables uniquely identifies $R(i, j-1)$.

6.5 Experimental Results

In this section we show the effectiveness of our approach. A single problem is considered and the period demands are generated from seasonal data with no trend: $\tilde{d}_t = 50[1 + \sin(\pi t/6)]$. In addition to the "no trend" case (P1) we also consider three others:

(P2) positive trend case, $\tilde{d}_t = 50[1 + \sin(\pi t/6)] + t$

(P3) negative trend case, $\tilde{d}_t = 50[1 + \sin(\pi t/6)] + (52 - t)$

(P4) life-cycle trend case, $\tilde{d}_t = 50[1 + \sin(\pi t/6)] + \min(t, 52 - t)$

In each test we assume an initial null inventory level and a normally distributed demand for every period with a coefficient of variation $\sigma_t/\tilde{d}_t$ for each $t \in \{1, \ldots, N\}$, where N is the length of the considered planning horizon. We performed tests using four different ordering cost values $a \in \{50, 100, 150, 200\}$ and two different $\sigma_t/\tilde{d}_t \in \{1/3, 1/6\}$. The planning horizon length takes even values in the range $[20, 38]$. The holding cost used in these tests is $h = 1$ per unit per period. Our tests also consider two different shortage cost values $s = 15$ and $s = 25$. Direct item cost is $v = 2$ per unit produced. All the experiments were performed on an Intel(R) Centrino(TM) CPU 1.50GHz with 500Mb RAM. The solver used is Choco [58], an open-source solver developed in Java. The cost-based filtering techniques presented are implemented as dedicated constraints within Choco. The same variable and value selection heuristics used in (Tarim et al. [87]) are employed. Tables 6.4 and 6.5 show the performance (in seconds) of our CP model enhanced with the cost-based filtering described in the former section. In our test results "$--$" means that within the time limit of 5 seconds the CP approach could not find an optimal solution. When the cost-based filtering method we proposed is not used, the pure CP approach is never able to provide an optimal solution within the given running time limit for every instance. Finally it should be also noted that the worst case running time of our approach over the whole test bed was $6,77$ minutes. Therefore even in the few cases where an optimal solution is not found in a less than a second, our cost-based filtering techniques provides a reasonable running time.

6.6 Conclusions

We presented a CP approach that finds optimal (R_n,S_n) policies under non-stationary demands. Using our approach it is now possible to evaluate the quality of a previously published MIP-based approximation method. Using a set of problem instances we showed that a piecewise approximation with seven segments usually provides good quality solutions, while using only two segments can yield solutions that differ significantly from the optimal. Furthermore we exploited convexity of the cost function to dynamically generate bounds during the search. The cost-based filtering technique we presented is able to speed up the search for opti-

		Test Set P1				Test Set P2			
		$\sigma_t/\tilde{d}_t = 1/3$		$\sigma_t/\tilde{d}_t = 1/6$		$\sigma_t/\tilde{d}_t = 1/3$		$\sigma_t/\tilde{d}_t = 1/6$	
a	N	$s = 15$	$s = 25$	$s = 15$	$s = 25$	$s = 15$	$s = 25$	$s = 15$	$s = 25$
50	20	0,150	0,030	0,020	0,020	0,030	0,040	0,050	0,050
	22	−−	−−	0,020	0,030	0,040	0,030	0,060	0,060
	24	0,031	0,040	0,030	0,030	0,060	0,040	0,080	0,070
	26	0,040	0,070	0,040	0,040	0,060	0,050	0,120	0,120
	28	0,050	0,080	0,060	0,050	0,070	0,060	0,170	0,121
	30	0,080	0,090	0,060	0,050	0,080	0,081	0,161	0,161
	32	0,100	0,090	0,070	0,081	0,120	0,141	0,180	0,150
	34	−−	−−	0,060	0,070	0,140	0,080	0,180	0,160
	36	0,210	0,111	0,080	0,081	0,161	0,090	0,230	0,180
	38	0,171	0,100	0,090	0,080	0,140	0,120	0,210	0,241
100	20	0,030	5,949	0,020	0,030	0,040	0,030	0,020	0,020
	22	0,030	0,030	0,030	0,030	0,040	0,030	0,031	0,030
	24	0,030	0,040	0,040	0,030	0,040	0,041	0,040	0,030
	26	0,040	0,040	0,040	0,040	0,080	0,050	0,050	0,050
	28	0,060	0,070	0,050	0,050	0,060	0,071	0,060	0,051
	30	0,061	0,060	0,060	0,060	0,071	0,080	0,061	0,080
	32	0,080	−−	0,070	0,070	0,081	0,090	0,071	0,070
	34	0,070	0,060	0,070	0,070	0,090	0,080	0,231	0,070
	36	0,080	0,101	0,071	0,071	0,101	0,100	0,090	0,090
	38	0,080	0,101	0,090	0,091	0,110	0,120	0,100	0,101
150	20	0,020	0,020	0,030	0,021	0,030	0,020	0,020	0,030
	22	0,030	0,030	0,030	0,020	0,030	0,030	0,030	0,030
	24	0,040	0,040	0,030	0,030	0,040	0,040	0,040	0,030
	26	0,040	0,040	0,040	0,040	0,040	0,050	0,050	0,061
	28	0,050	0,050	0,050	0,041	0,060	0,061	0,050	0,050
	30	0,070	0,071	0,050	0,061	0,070	0,070	0,060	0,070
	32	0,070	4,306	0,060	0,071	0,080	0,080	0,070	0,070
	34	0,070	0,070	0,060	0,070	0,100	0,080	0,070	0,071
	36	0,080	0,080	0,070	0,080	0,090	0,110	0,080	0,090
	38	0,090	0,100	0,100	0,080	0,110	0,120	0,110	0,121
200	20	0,030	0,030	0,030	0,020	0,031	0,040	0,030	0,020
	22	0,030	0,220	0,030	0,030	0,030	0,041	0,030	0,030
	24	0,030	0,040	0,030	0,040	0,040	0,040	0,030	0,041
	26	0,040	0,040	0,040	0,040	0,050	0,051	0,041	0,050
	28	0,050	0,050	0,051	0,060	0,080	0,060	0,060	0,050
	30	0,070	0,060	0,060	0,060	0,070	0,070	0,070	0,070
	32	0,080	0,080	0,060	0,060	0,080	0,090	0,070	0,070
	34	0,070	−−	0,070	0,070	0,090	0,080	0,080	0,081
	36	0,080	0,081	0,070	0,070	0,110	0,101	0,090	0,110
	38	0,100	0,090	0,091	0,090	0,121	0,100	0,110	0,110

Table 6.4: Test Set P1, P2.

mal (R_n, S_n) policy parameters under a shortage cost scheme. Our experimental results prove that such a technique brings a significant improvement in the efficiency of the pure CP approach for this problem. We are now able to solve problems over a planning horizon up to forty periods, typically in a fraction of a second and in the worst case in a few minutes. This means that our approach can be now applied to problems of a realistic size.

		Test Set P3				Test Set P4			
		$\sigma_t/\tilde{d}_t = 1/3$		$\sigma_t/\tilde{d}_t = 1/6$		$\sigma_t/\tilde{d}_t = 1/3$		$\sigma_t/\tilde{d}_t = 1/6$	
a	N	$s = 15$	$s = 25$	$s = 15$	$s = 25$	$s = 15$	$s = 25$	$s = 15$	$s = 25$
50	20	0,321	0,170	0,330	0,160	0,070	0,030	0,050	0,061
	22	0,480	0,300	0,370	0,341	0,030	0,040	0,060	0,060
	24	0,581	0,310	0,531	0,421	0,050	0,040	0,110	0,071
	26	1,222	0,501	0,791	0,531	0,070	0,060	0,090	0,090
	28	2,224	0,661	1,142	0,741	0,140	0,070	0,120	0,160
	30	2,013	0,722	1,052	0,751	0,100	0,060	0,130	0,170
	32	1,812	0,941	1,182	0,801	0,121	0,080	0,180	0,140
	34	1,883	0,862	1,312	0,952	0,120	0,090	0,190	0,150
	36	2,093	0,981	1,472	1,152	0,121	0,110	0,210	0,180
	38	3,636	1,131	1,803	1,512	0,120	0,100	0,251	0,200
100	20	0,030	0,040	0,060	0,070	0,040	0,030	0,030	0,020
	22	0,040	0,040	0,070	0,071	0,040	0,030	0,030	0,030
	24	0,040	0,050	0,090	0,080	0,050	0,030	0,040	0,040
	26	0,050	0,281	0,100	0,100	0,050	0,050	0,050	0,040
	28	0,070	0,070	0,131	0,120	0,061	0,060	0,060	0,060
	30	0,070	0,070	0,140	0,130	0,070	0,070	0,070	0,060
	32	0,080	0,080	0,150	0,160	0,080	0,080	0,070	0,070
	34	0,090	0,090	0,161	0,210	0,090	0,081	0,080	0,070
	36	0,100	0,110	0,240	0,180	0,090	0,090	0,090	0,080
	38	0,141	0,130	0,211	0,250	0,100	0,100	0,110	0,100
150	20	0,040	0,030	0,060	0,060	0,030	0,030	0,030	0,030
	22	0,040	0,040	0,071	0,070	0,030	0,030	0,030	0,030
	24	0,050	0,041	0,140	0,080	0,040	0,030	0,040	0,030
	26	0,060	0,050	0,100	0,160	0,060	0,040	0,050	0,040
	28	0,070	0,070	0,120	0,170	0,060	0,060	0,060	0,050
	30	0,070	0,070	0,130	0,140	0,070	0,060	0,070	0,060
	32	0,090	0,090	0,160	0,220	0,080	0,080	0,070	0,070
	34	0,091	0,090	0,171	0,170	0,080	0,090	0,080	0,080
	36	0,100	0,110	0,181	0,250	0,100	0,100	0,090	0,090
	38	0,140	0,120	0,220	0,260	0,100	0,101	0,100	0,100
200	20	0,071	0,030	0,060	0,070	0,040	0,030	0,030	0,030
	22	0,090	0,070	0,070	0,120	0,030	0,030	0,030	0,030
	24	0,090	0,130	0,080	0,080	0,050	0,040	0,030	0,040
	26	0,110	0,110	0,100	0,171	0,050	0,050	0,051	0,050
	28	0,130	0,170	0,181	0,130	0,070	0,070	0,060	0,050
	30	0,210	0,150	0,150	0,151	0,060	0,070	0,070	0,060
	32	0,210	0,090	0,150	0,221	0,070	0,070	0,070	0,080
	34	0,210	0,241	0,180	0,160	0,080	0,080	0,080	0,081
	36	0,250	0,210	0,241	0,190	0,090	0,100	0,080	0,090
	38	0,221	0,271	0,260	0,210	0,140	0,110	0,100	0,131

Table 6.5: Test Set P3, P4.

Bibliography

[1] K. Apt. *Principles of Constraint Programming*. Cambridge University Press, Cambridge, UK, 2003.

[2] K. J. Arrow, T. Harris, and J. Marshcak. Optimal inventory policy. *Econometrica*, 53(6):250–272, 1951.

[3] R. G. Askin. A procedure for production lot sizing with probabilistic dynamic demand. *AIIE Transactions*, 13:132–137, 1981.

[4] M. Avriel and A. C. Williams. The value of information and stochastic programming. *Operations Research*, 18(5):947–954, 1970.

[5] T. Balafoutis and K. Stergiou. Algorithms for stochastic csps. In Frédéric Benhamou, editor, *Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings*, volume 4204 of *Lecture Notes in Computer Science*, pages 44–58. Springer, 2006.

[6] S. Bashyam and M.C. Fu. Optimization of (s,S) inventory systems with random lead times and a service level constraint. *Management Science*, 44(12):243–256, 1998.

[7] J. C. Beck and L. Perron. Discrepancy-bounded depth first search. 2000.

[8] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[9] T. Benoist, E. Bourreau, Y. Caseau, and B. Rottembourg. Towards stochastic constraint programming: A study of online multi-choice knapsack with

deadlines. In Toby Walsh, editor, *Principles and Practice of Constraint Programming - CP 2001, 7th International Conference, CP 2001, Paphos, Cyprus, November 26 - December 1, 2001, Proceedings*, volume 2239 of *Lecture Notes in Computer Science*, pages 61–76. Springer, 2001.

[10] W. L. Berry. Lot sizing procedures for requirements planning systems: A framework for analysis. *Production and Inventory Management Journal*, 13:19–34, 1972.

[11] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer Verlag, New York, 1997.

[12] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and Valued CSPs: Basic Properties and Comparison. In M. Jampel, E. Freuder, and M. Maher, editors, *Over-Constrained Systems (Selected papers from the Workshop on Over-Constrained Systems at CP'95, reprints and background papers)*, volume 1106 of *Lecture Notes in Computer Science*, pages 111–150. Springer, 1996.

[13] S. Bistarelli, U. Montanari, and F. Rossi. Soft constraint logic programming and generalized shortest path problems. *Journal of Heuristics, Kluwer*, 2001.

[14] G. R. Bitran and H. H. Yanasse. Computational complexity of the capacitated lot size problem. *Management Science*, 28(10):1174–1186, 1982.

[15] J. H. Bookbinder and J. Y. Tan. Strategies for the probabilistic lot-sizing problem with service-level constraints. *Management Science*, 34:1096–1108, 1988.

[16] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[17] S. C. Brailsford, C. N. Potts, and B. M. Smith. Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research*, 119:557–581, 1999.

[18] A. Charnes and W. W. Cooper. Chance-constrainted programming. *Management Science*, 6(1):73–79, 1959.

[19] A. Charnes and W. W. Cooper. Deterministic equivalents for optimizing and satisficing under chance constraints. *Operations Research*, 11(1):18–39, 1963.

[20] N. Christofides, A. Mingozzi, and P. Toth. State space relaxation procedures for the computation of bounds to routing problems. *Networks*, 11:145–164, 1981.

[21] T. Davis. Effective supply chain management. Sloan Management Review, 1993.

[22] A. G. de Kok. Basics of inventory management: part 2 The (R,S)-model. Research memorandum, FEW 521, 1991. Department of Economics, Tilburg University, Tilburg, The Netherlands.

[23] T. de Kok and K. Inderfurth. Nervousness in inventory management: Comparison of basic control rules. *European Journal of Operational Research*, 103:55–82, 1997.

[24] J. L. Devore. *Probability and Statistics for Engineering and the Sciences*. Duxbury Press, 1995. Fourth Edition.

[25] S. B. Dreyfus and A. M. Law. *The Art And Theory of Dynamic Programming*. Academic Press, 1989.

[26] F. Edgeworth. The mathematical theory of banking. *Journal of the Royal Statistical Society*, 51:113–127, 1888.

[27] G. D. Eppen and R. K. Martin. Determining safety stock in the presence of stochastic lead time and demand. *Management Science*, 34:1380–1390, 1988.

[28] T. Fahle and M. Sellmann. Cost-based filtering for the constrained knapsack problem. *Annals of Operations Research*, 115:73–93, 2002.

[29] H. Fargier, J. Lang, R. Martin-Clouaire, and T. Schiex. A constraint satisfaction framework for decision under uncertainty. In Philippe Besnard and Steve Hanks, editors, *UAI '95: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, August 18-20, 1995, Montreal, Quebec, Canada*, pages 167–174. Morgan Kaufmann, 1995.

[30] M. Florian, J. K. Lenstra, and A. H. G. Rinooy Kan. Deterministic production planning: Algorithms and complexity. *Management Science*, 26(7):669–679, 1980.

[31] F. Focacci, A. Lodi, and M. Milano. Cost-based domain filtering. In Joxan Jaffar, editor, *Principles and Practice of Constraint Programming - CP'99, 5th International Conference, Alexandria, Virginia, USA, October 11-14, 1999, Proceedings*, volume 1713 of *Lecture Notes in Computer Science*, pages 189–203. Springer, 1999.

[32] F. Focacci, A. Lodi, and M. Milano. Optimization-oriented global constraints. *Constraints*, 7(3-4):351–365, 2002.

[33] F. Focacci and M. Milano. Connections and integrations of dynamic programming and constraint programming. In *Proceedings of the Third International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CPAIOR 2001*, 2001.

[34] L. Fortuin. Five popular probability density functions: a comparison in the field of stock-control models. *Journal of the Operational Research Society*, 31(10):937–942, 1980.

[35] D. W. Fowler and K. N. Brown. Branching constraint satisfaction problems and markov decision problems compared. *Annals of Operations Research*, 118:85–100, 2003.

[36] E. C. Freuder and R. J. Wallace. Partial constraint satisfaction. *Artif. Intell.*, 58(1-3):21–70, 1992.

[37] M. R. Garey and D. S. Johnson. *Computer and Intractability. A guide to the theory of NP-Completeness*. Bell Laboratories, Murray Hill, New Jersey, 1979.

[38] M. L. Ginsberg and W. D. Harvey. Iterative broadening. *Artif. Intell.*, 55(2-3):367–383, 1992.

[39] H.-J. Girlich and A. Chikan. The origins of dynamic inventory modelling under uncertainty - (the men, their work and connection with the stanford studies). *International Journal of Production Economics*, 71(1):351–363, 2001.

[40] S. C. Graves. A Single-Item Inventory Model for a Non-Stationary Demand Process. *Manufacturing & Service Operations Management*, 1:50–61, 1999.

[41] G. Hadley and T. M. Whitin. *Analysis of Inventory Systems*. Prentice Hall, 1964.

[42] M. Hariga and M. Ben Daya. Some stochastic inventory models with deterministic variable lead time. *European Journal of Operational Research*, 113:42–51, 1999.

[43] J. C. Hayya, S. H. Xu, R. V. Ramasesh, and X. X. He. Order crossover in inventory systems. *Stochastic Models*, 11(2):279–309, 1995.

[44] G. Heisig. *Planning Stability in Material Requirements Planning Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[45] P. Van Hentenryck and J. P. Carillon. Generality vs. specificity: an experience with ai and or techniques. In *Proceedings of the 7th National Conference on Artificial Intelligence. (AAAI-88) St. Paul, MN, August 21-26*, pages 660–664, 1988.

[46] P. Van Hentenryck, L. Michel, L. Perron, and J.-C. Régin. Constraint programming in opl. In Gopalan Nadathur, editor, *Proceedings of the International Conference on Principles and Practice of Declarative Programming*

(PPDP'99), volume 1702 of *Lecture Notes in Computer Science*, pages 98–116, September 29 - October 1 1999.

[47] J. N. Hooker, G. Ottosson, E. S. Thorsteinsson, and H. J. Kim. On integrating constraint propagation and linear programming for combinatorial optimization. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 136–141. AAAI, The AAAI Press/MIT Press, Cambridge, Ma., 1999.

[48] J. A. Hunt. Balancing accuracy and simplicity in determining reorder points. *Management Science*, 12(4):B94–B103, 1965.

[49] Inc. ILOG. *OPL Studio 3.7 Users Manual*. ILOG, Inc., Incline Village, NV, 2007.

[50] V. Jain and I. E. Grossmann. Algorithms for hybrid milp/cp models for a class of optimization problems. *INFORMS Journal on computing*, 13:258–276, 2001.

[51] F. Janssen and T. de Kok. A two-supplier inventory model. *International Journal of Production Economics*, 59:395–403, 1999.

[52] H. Jeffreys. *Theory of Probability*. Clarendon Press, Oxford, UK, 1961.

[53] L. A. Johnson and D. C. Montgomery. *Operations Research in Production Planning, Scheduling, and Inventory Control*. Wiley, New York, 1974.

[54] P. Kall and S. W. Wallace. *Stochastic Programming*. John Wiley & Sons, 1994.

[55] R. S. Kaplan. A dynamic inventory model with stochastic lead times. *Management Science*, 17(7):491–507, 1970.

[56] A. J. Kleywegt, A. Shapiro, and T. Homem-De-Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal of Optimization*, 12(2):479–502, 2001.

[57] D. Kuhn. *Generalized bounds for convex multistage stochastic programs*, volume 584 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, 2005.

[58] F. Laburthe and the OCRE project team. Choco: Implementing a cp kernel. Technical report, Bouygues e-Lab, France, 1994.

[59] R. Levi, R. O. Roundy, and D. B. Shmoys. Provably near-optimal sampling-based algorithms for stochastic inventory control models. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 739–748, New York, NY, USA, 2006. ACM Press.

[60] M. L. Littman, J. Goldsmith, and M. Mundhenk. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9:1–36, 1998.

[61] M. L. Littman, S. M. Majercik, and T. Pitassi. Stochastic boolean satisfiability. *J. Autom. Reason.*, 27(3):251–296, 2001.

[62] I. J. Lustig and J. F. Puget. Program does not equal program: Constraint programming and its relationship to mathematical programming. *Interfaces*, 31:29–53, 2001.

[63] S. Martello and P. Toth. *Knapsack Problems*. John Wiley & Sons, NY, 1990.

[64] E. L. Porteus. *Foundations of Stochastic Inventory Theory*. Stanford University Press, Stanford, CA, 2002.

[65] I. N. Pujawan and E. A. Silver. Augmenting the lot sizing order quantity when demand is probabilistic. *European Journal of Operational Research*, 127(3):705–722, August 2008.

[66] J.-C. Regin. A filtering algorithm for constraints of difference in csps. In *Proceedings of the 12th National Conference on Artificial Intelligence, Volume 1. Seattle, WA, USA, July 31 - August 4*, pages 362–367. AAAI Press, 1994.

[67] J.-C Regin. *Global Constraints and Filtering Algorithms*. in Constraints and Integer Programming Combined, Kluwer, M. Milano editor, 2003.

[68] J. Riezebos. Inventory order crossovers. *International Journal of Production Economics*, 104(2):666–675, 2006.

[69] J. Riezebos and G. J. C. Gaalman. Modeling expected inventory order crossovers. In *Pre-prints of the Fourteenth International Working Seminar on Production Economics*, pages 4:137–148, 2006. Innsbruck, Austria, February 20–24.

[70] F. Rossi, C. Petrie, and V. Dhar. On the equivalence of constraint satisfaction problems. In *European Conference on Artificial Intelligence*, pages 550–556, 1990. Stockholm, Sweden.

[71] R. Rossi, S. A. Tarim, B. Hnich, and S. Prestwich. Replenishment planning for stochastic inventory systems with shortage cost. In Pascal Van Hentenryck and Laurence A. Wolsey, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 4th International Conference, CPAIOR 2007, Brussels, Belgium, May 23-26, 2007, Proceedings*, volume 4510 of *Lecture Notes in Computer Science*, pages 229–243. Springer Verlag, 2007.

[72] R. Rossi, S. A. Tarim, B. Hnich, and S. Prestwich. Computing Replenishment Cycle Policy under Non-stationary Stochastic Lead Time. *International Journal of Production Economics*, 2008. submitted for possible publication.

[73] R. Rossi, S. A. Tarim, B. Hnich, and S. Prestwich. Constraint Programming for Stochastic Inventory Systems under Shortage Cost. *European Journal of Operational Research*, 2008. submitted for possible publication.

[74] R. Rossi, S. A. Tarim, B. Hnich, and S. Prestwich. Cost-based filtering for stochastic constraint programming. In *Proceedings of The 14th International Conference on Principle and Practice of Constraint Programming (CP-2008), Sep. 14-18, 2008 - Sydney, Australia*, volume 5202 of *Lecture Notes in Computer Science*, pages 235–250. Springer Verlag, 2008.

[75] R. Rossi, S. A. Tarim, B. Hnich, and S. Prestwich. A global chance-constraint for stochastic inventory systems under service level constraints. *Constraints*, 13(4):xxx–xxx, 2008.

[76] H. Scarf. *The optimality of (s,S) policies in dynamic inventory problem*. in Arrow, Karlin and Suppes, Mathematical Methods in Social Sciences 1959, Stanford Unversity Press, 1960.

[77] R. Sedgewick. *Algorithms (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1988.

[78] J. K. Sengupta. *Stochastic Programming: Methods and Applications*. North-Holland, Amsterdam, 1972.

[79] N. M. Shazeer, M. L. Littman, and G. A. Keim. Solving crossword puzzles as probabilistic constraint satisfaction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence, July 18-22, Orlando, Florida, USA*, pages 156–162, 1999.

[80] E. A. Silver. Inventory control under a probabilistic time-varying demand pattern. *AIIE Transactions*, 10:371–379, 1978.

[81] E. A. Silver, D. F. Pyke, and R. Peterson. *Inventory Management and Production Planning and Scheduling*. John-Wiley and Sons, New York, 1998.

[82] S. R. Smits, M. Wagner, and T. G de Kok. Determination of an order-up-to policy in the stochastic economic lot scheduling model. *International Journal of Production Economics*, 90:377–389, 2004.

[83] C. R. Sox. Dynamic lot sizing with random demand and non-stationary costs. *Operations Research Letters*, 20:155–164(10), May 1997.

[84] M. L. Stein. Large sample properties of simulation using latin hypercube sampling. *Technometrics*, 29:143–151, 1987.

[85] C. S. Tang. Perpectives in supply chain risk management. *International Journal of Production Economics*, 103:451–488, 2006.

[86] S. A. Tarim. *Dynamic Lotsizing Models for Stochastic Demand in Single and Multi-Echelon Inventory Systems*. PhD thesis, Lancaster University, 1996.

[87] S. A. Tarim, B. Hnich, R. Rossi, and S. Prestwich. Cost-based filtering for stochastic inventory control. In *Recent Advances in Constraints Joint ERCIM/CoLogNET International Workshop on Constraint Solving and Constraint Logic Programming, CSCLP 2006*, pages 169–183. Springer Verlag, 2007. Lecture Notes in Artificial Intelligence No. 4651.

[88] S. A. Tarim, B. Hnich, R. Rossi, and S. Prestwich. Cost-based filtering techniques for stochastic inventory control under service level constraints. *Constraints*, x(x):xxx–xxx, 2009.

[89] S. A. Tarim and B. G. Kingsman. The stochastic dynamic production/inventory lot-sizing problem with service-level constraints. *International Journal of Production Economics*, 88:105–119, 2004.

[90] S. A. Tarim and B. G. Kingsman. Modelling and Computing (R^n,S^n) Policies for Inventory Systems with Non-Stationary Stochastic Demand. *European Journal of Operational Research*, 174:581–599, 2006.

[91] S. A. Tarim, S. Manandhar, and T. Walsh. Stochastic constraint programming: A scenario-based approach. *Constraints*, 11(1):53–80, 2006.

[92] S. A. Tarim and B. Smith. Constraint Programming for Computing Non-Stationary (R,S) Inventory Policies. *European Journal of Operational Research*, 189:1004–1021, 2008.

[93] H. Tempelmeier. On the stochastic uncapacitated dynamic single-item lot-sizing problem with service level constraints. *European Journal of Operational Research*, 127(1):184–194, August 2007.

[94] E. S. Thorsteinsson. *Hybrid Approaches to Combinatorial Optimisation*. PhD thesis, Carnegie Mellon University, Schenley Park, Pittsburgh PA 15213, U.S.A., May 2001.

[95] S. Vajda. *Probabilistic Programming*. Academic Press, New York, 1972.

[96] H. M. Wagner and T. M. Whitin. Dynamic version of the economic lot size model. *Management Science*, 5:89–96, 1958.

[97] T. Walsh. Depth-bounded discrepancy search. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97, Nagoya, Japan, August 23-29*, pages 1388–1395. Morgan Kaufmann, 1997.

[98] T. Walsh. Stochastic constraint programming. In Frank van Harmelen, editor, *Proceedings of the 15th Eureopean Conference on Artificial Intelligence, ECAI'2002, Lyon, France, July 2002*, pages 111–115. IOS Press, 2002.

[99] J.-P. Watson and J. C. Beck. A hybrid constraint programming / local search approach to the job-shop scheduling problem. In Laurent Perron and Michael A. Trick, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 5th International Conference, CPAIOR 2008, Paris, France, May 20-23, 2008, Proceedings*, volume 5015 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 2008.

[100] C. A. Yano and H. L. Lee. Lot sizing with random yields: A review. *Operations Research*, 43(2):311–334, 1995.

[101] P. H. Zipkin. Stochastic leadtimes in continuous-time inventory models. *Naval Research Logistics Quarterly*, 33:763–774, 1986.

[102] P. H. Zipkin. *Foundations of Inventory Management*. McGraw-Hill/Irwin, Boston, Mass., 2000.